2012.05

PROGRAMMER 程序员

27 美国云计算印象

78 自己想办法
有关工程现状的几点反思

82 程序员的创新修炼

86 硅谷产品管理大师 Marty Cagan
如何获取用户需求

90 怎样营造良好的技术文化

93 青蛙设计首席创意执行官
Mark Rolston
我们所做的一切都是为了创新

104 B2G
来自Web平台的挑战者
由Mozilla和著名运营商Telefonica共同开发的新的操作系统——B2G，从诞生开始，就肩负着瓦解苹果和Google垄断地位的使命

106 狂奔的移动端用户体验设计

117 探索Duqu木马身世之谜

122 Ready? Go!
Go语言开发背景、语法和类型

135 AWS推动者
Werner Vogels

ISSN 1672-3252
9 771672 325128 05>
邮发代号：2-665　定价：15元
www.programmer.com.cn　www.csdn.net

Contents

P6

资讯

6 新闻

8 新产品新工具

10 外刊速递

13 程序天下事

16 报道

P26

封面报道：云计算！

本期封面报道，有来自美国硅谷云计算公司的一线报道，有典型互联网技术发展历程的精辟总结，有视频网站YouTube累积七年的可扩展经验，更有来自微软、盛大、新浪、淘宝、百度、腾讯、有道、爱奇艺、中科院的众位嘉宾披挂上阵，从IaaS、PaaS、SaaS三个层面全方位分享云计算技术一线的成果。

27 美国云计算印象

32 可伸缩性的10年探索
——知名网站的技术发展历程

36 YouTube成功秘诀
——七年可扩展经验分享

40 改变互联网的IaaS服务

44 开源云先锋OpenStack

46 PaaS——云计算的下个制高点？

51 利用SAE数据存储技术开发应用

55 从数据仓库系统对比看Hive发展前景

58 NoSQL持久化

62 腾讯的NoSQL应用实践

66 剖析同步云存储系统架构

70 淘宝网“双12”背后的技术故事

74 云计算与大数据挖掘

P20

管理

78 自己想办法

——有关工程现状的几点反思

从过程、方法和工具构成的工程理论，到人本思想为基础的敏捷实务，作者指出种种工程理论中最基础的组织因素不过是一事一职，并提出了以项目、团队与产品三种角色及其视角构成的具体化的工程思维方向。

82 程序员的创新修炼

以系统的方法来理解创新思维的基本方面有助于了解持续创新的内在规律。本文作者根据多年的工作体验和思考，展现出了一个循序渐进的创新思考模型，并结合实例进行了深入的阐释和分析。

86 如何获取用户需求

Marty Cagan是享有世界声誉的产品管理专家，曾经担任网景副总裁、eBay产品管理及设计高级副总裁。本文是他回顾自己二十多年来从事软件产品管理工作的总结和经验分享，分析了现有市场调研方法的利弊，并指出特约用户的重要性。

90 怎样营造良好的技术文化

众所周知，良好的技术文化不仅有助于人才成长和提升团队内聚力，更是吸引和留住优秀人才的一大法宝。然而，该如何营造良好的技术文化呢？且听本期三位嘉宾的经验分享。

移动

93 我们所做的一切都是为了创新

——青蛙设计首席创意执行官Mark Rolston专访

青蛙设计首席创意执行官Mark Rolston在接受《程序员》特约撰稿人西乔的采访时谈到，软件是设计的首要推动力，中国市场也正在意识到需要高品质的设计来帮助产品脱颖而出，并预测数据可视化将成为设计的重要分支，而数字世界和自然世界最终将合二为一。

P132

98 情感牵引和交互在游戏中的价值

本文详尽阐述了以关系链和情感沉浸为基础打造的情感牵引和交互在游戏中的价值，作者认为，捕获用户在情感线层面的投入已成为未来必需的层面。

101 Web App对企业移动开发的影响

作者认为，对于传统企业的移动开发，多开发平台带来的学习成本、人力成本以及开发团队的管理成本都不容小觑，而Web App能为其带来很好的解决方案。

104 B2G：来自Web平台的挑战者

iOS和Android已成为移动平台的两大帝国，而由Mozilla和著名运营商Telefonica共同开发的新的操作系统——B2G，从诞生开始，就肩负着瓦解苹果和Google垄断地位的使命。

106 狂奔的移动端用户体验设计（上）

本文讲述了在移动互联网时代，如何在多变的终端配置、网络环境以及混乱的网页内容下，设计出具有优秀用户体验的产品，并分享了在实际项目中的设计迭代经验。

2012年05月刊　总第235期

Contents

主管：中国社会科学院
主办：中国社会科学院文献信息中心
出版：《程序员》杂志社
网址：http://www.programmer.com.cn
国际刊号：ISSN 1672-3252
国内刊号：CN11-5038/G2
邮发代号：2-665
广告经营许可证号：京东工商广字0188号

总编：黄长著　Editor-in-chief：Huang Changzhu
社长/常务副总编：张悦校　President：Zhang Yuexiao
副社长：蒋 涛　Vice President：Jiang Tao
编委会：黄长著　张悦校　陈洋彬　蒋涛　曾登高　刘江
Editorial Member：Huang Changzhu　Zhang Yuexiao　Chen Yangbin
Jiang Tao　Zeng Denggao　Liu Jiang

执行总编：刘江　Executive Editor-in-chief：Liu Jiang
执行主编：孟迎霞　Managing Editor：Meng Yingxia
编辑部主任：董世晓　Director：Dong Shixiao
责任编辑：陈博　杨爽　卢鹄翔
Editors：Chen Bo　Yang Shuang　Lu Dongxiang
特邀编辑：蔡学镛　池建强　冯大辉　高博　肖新光　杨栋　余晟　周爱民
Contributing Editors：Cai Xueyong　Chi Jianqiang　Feng Dahui　Gao Bo
Xiao Xinguang　Yang Dong　Yu Sheng　Zhou Aimin
美术设计：纪明超　Art Designer：Ji Mingchao
美术编辑：朱凯　Art Editor：Zhu Kai
流程编辑：白羽中　Coordinator：Bai Yuzhong
Tel：010-64351458
E-mail：editor@csdn.net

广告总代理：北京创新乐知信息技术有限公司
Sole Advertising Agency：Bejing CSDN Co.,Ltd
Tel：010-64376055
E-mail：ad@csdn.net
Marketing Dept：010-51661202（ext 149）
E-mail：market@csdn.net

发行部
Distribution Dept.010-64351431
E-mail：sales@csdn.net

读者服务部
Readers service Dept.
网上订购：http://dingyue.programmer.com.cn/
读者信箱：reader@csdn.net
地址：北京市朝阳区广顺北大街33号院1号楼福码大厦B座12层
Address: B-12th Floor Fairmont Tower NO.33 Guangshun North street,Chaoyang District,Beijing
邮政编码：100102
电话：010-64351436
传真：010-64348545

法律顾问：北京中润律师事务所 王杰
Law Consultant：Beijng Zhongrun Lawyer Firm
印刷：北京盛通印刷股份有限公司
Print: Beijing Shengtong Printing Co., Ltd.
出版日期：每月1日
Publication Date：the first day per month
零售价：RMB 15.00元　新台币 390元　HK $ 35.00（港、澳）
US $ 9.00（海外）
Retail Price：RMB 15, NT$390, HK $ 35.00,US $ 9.00

110 Android系统换肤功能的设计思路
作者在文中提出了直接读取外部资源文件、PackageManager、重定向资源ID、重定向资源包路径以及重定向资源文件路径这5种实现Android系统换肤功能的设计思路，并对它们的优缺点进行了比较。

技术

117 探索Duqu木马身世之谜
——Duqu和Stuxnet同源性分析
同源，是生物遗传学领域的重要概念，用于描述物种或DNA序列是否具有相同的祖先。计算机病毒分析工程师也经常通过分析病毒同源性的方法，追踪病毒制造者的身份——比如dvlodr（口令蠕虫）和lovegate（爱门蠕虫）的制造者，就是通过同源性分析判断为同一人。

122 Ready? Go!
——Go语言开发背景、语法和类型
Go语言是Google于2009年推出的静态编译型语言，旨在为开发人员提供类似Python、Ruby一样简洁的语言环境，同时又具备C一样的运行效率。Go以社区协作的形式，不断完善语言和标准库的设计与实现，终于在2012年3月28日发布了第一个稳定的发行版本：Go 1。

126 Windows 8的Metro应用（下）
前两期我们分别介绍了Metro应用的基本特性和支撑Metro应用的新API系统—WinRT。本期我们将介绍如何亲自动手编写一个新的Metro应用，从编译器和编程语言的角度诠释Metro应用。

百味

130 新书上架

132 Geek

135 AWS推动者 Werner Vogels

136 程序幽默

企业专栏

20 探究中国化的云计算落地之路

22 基于云端架构提升移动阅读体验
——兼谈UC小说阅读器

移动

4月10日，社交网络巨头Facebook宣布以10亿美元现金加股票的方式收购移动照片分享应用Instagram。Instagram CEO Kevin Systrom发表声明表示，收购之后的Instagram将继续独立运营，但与Facebook之间会加强合作和联系。Facebook CEO扎克伯格也在自己的Facebook页面上发布声明表示已经达成收购意向。

2011年Facebook曾推出过自己的移动照片分享应用程序，但反响平平。而这次，Facebook期望能够通过收购Instagram直接获取大量的摄影师以及拍照爱好者，同时能够迅速提升自身在照片分享领域的影响力。

有消息透露，在过去的一年半时间内，Google和Twitter都曾与Instagram进行过多次接触，但都没有达成交易。

在此之前的一周，Instagram推出了Android版本，仅24小时下载量就超过了100万，而在推出Android版本之后，Instagram的用户数可能将会迅速增加至5000万。这或许也是Facebook决定收购Instagram的原因。

Windows 8

4月17日，微软Windows官方博客发表文章，公布了Windows 8的最新版本信息。Windows 8将推出3个版本，分别是：Windows 8、Widnows 8 Pro和Windows RT。

对大多数个人用户来说，Windows 8是首选。它包括基础的最新版Windows资源管理器、任务管理器、更好的多显示器支持，以及一些之前仅在Windows企业版和旗舰版中才包含的功能。

Windows 8 Pro是专门为技术爱好者以及企业、科研机构人员提供的，除了基础的Windows 8功能之外，Windows 8 Pro还将为这些用户提供更多的技术特性。其中包括加密、虚拟化、计算机管理以及域连通等功能。而Windows媒体中心这类媒体应用将会以更简约的“媒体包（media pack）”扩展形式出现在Windows 8 Pro中。

Windows RT是Windows家族的新成员，也就是我们之前所说的Windows ARM版或者WOA版本。Windows RT仅会预装在使用ARM处理器的PC或者平板电脑上，而不会像其他版本一样公开发售。Windows RT中将会预装针对触屏操作优化过的新版Word、Excel、PowerPoint及OneNote软件。

安全

苹果在对Mac OS X的宣传中，一直都将“不会感染病毒”作为广告语之一，但Flashback木马的出现则让这条宣传语显得有些尴尬。

Flashback特洛伊木马在全球范围内创建了一个由逾60万台Mac组成的僵尸网络，收集用户个人资料。在用户访问恶意网站时，Flashback能利用一个Java安全缺陷，悄悄潜入用户计算机中。这是苹果Mac OS X操作系统有史以来最大规模的恶意软件感染事件。在全球67万被该木马感染的电脑中，98%使用Mac OS X。

卡巴斯基首席安全专家Alexander Gostev表示，苹果几月前就发现了这一威胁，但并未采取足够措施保护用户，而其他系统几个月前都已经采用了Oracle提供的Java补丁。

针对此次出现的感染事件，苹果发布一款Java安全升级包，用于清除Flashback特洛伊木马。据苹果称，Java升级包能清除“大多数常见的Flashback恶意件变种”，通过关闭Java applets自动执行功能，能预防未来的Flashback变种。

“微软现有的产品部门将继续参与到开源项目的建设和标准制订中，因为开源项目在今天变得越发重要。”

——4月13日，微软宣布将成立一家名为“微软开放技术”的公司，以推动开源项目的建设和标准的制订。新公司将由微软高管Jean Paoli担任总裁，并且公司也将会在发展中逐步推进和成立开源战略团队。

“我们在调查期间一直都很真诚地回答和配合。虽然被罚，但FCC对Google调查已经结束。”

——2010年Google曾涉嫌在拍摄街景的过程中同时收集未加密的WiFi数据，此举违反了美国联邦政府的相关法律。两年后，由于针对这起案件的最后法律有限期即将来临，FCC最终于作出罚款决定。

“Google已经开始注意到支持MIPS架构的平板出货量，我目前主要的工作是确保所有工具链做好准备，我们一定会得到Android官方对MIPS的支持。”

——MIPS移动工程师Amit Rohatgi在Linley移动峰会上透露，Google将同意Android支持MIPS架构。这个消息是随MIPS财务状况不佳，可能考虑降低售价的报道中传播出来的。MIPS架构有坚实的基础网络和机顶盒市场，但在移动终端市场仍然不是ARM的对手。

“今天是Stratasys里程碑式的日子，我们的合并将会促进3D打印的发展，我们也为数字制造业做出了榜样。”

——3D打印的两家领先企业Stratasys和Objet于4月17日宣布进行合并，交易额为14亿美元，合并后的公司名仍为Stratasys。此项合并将会进一步确立Stratasys在高速发展的3D打印及数字制造业中的领导地位。

“我们仍有许多事情要去完成。但这个成绩对公司而言是一个里程碑。”

——雅虎发言人表示，得益于搜索广告收入好于预期，雅虎实现了自2008年以来的首次净利增长。2012年第一季度，雅虎营收12.2亿美元，比去年同期增长了1%。除去成本花费，第一季度净利润为2.86亿美元，同比上涨了28%。

“坐在你身边的人极有可能真是以后对你有帮助的人，不管是在你寻求帮助之前或之后，与他们聊天也将成为创业经历中很有价值的一部分。”

——Instagram联合创始人Kevin Systrom在斯坦福大学的演讲中谈到自己与众多硅谷人的关系时如是说。从开始构思产品到被收购，整个过程都与硅谷那些响当当的人物有着密不可分的关系，而他的整个职业生涯也与社交网络有着紧密的联系。

CSDN十大资讯

2012年3月26日–4月26日

01 DuckDuckGo搜索引擎快速崛起

作者 / 张勇

DuckDuckGo搜索引擎近来发展迅猛，近3个月的搜索请求以平均每天227%的速度增长。这巨大的上升势头，一方面来自2011年1月开始的视觉界面重新设计，另一方面也是数据隐私日活动的鞭策。正是因为DuckDuckGo对数据隐私的重视，得到了很多用户尤其是黑客的青睐。

02 Google愚人节推出NES地图

作者 / 张勇

Google在地图产品中提供了各种系统接口，却一直忽略了广受欢迎的NES游戏系统。趁着愚人节的机会，Googel“调皮”地推出了NES 8位地图产品。此外继2011年推出的新功能“穿越搜索”后，Google也在这天发布了“水下搜索”。

03 维基百科弃用Google地图

作者 / 魏兵

维基百科最新版本的移动设备应用程序宣布放弃Google地图，转而采用开源地图服务OpenStreetMap。与苹果和Foursquare不同，维基百科原本可以靠非营利性身份获得免费使用Google地图服务的资格，但维基百科更倾向在服务中去除Google的私有API。

04 Web Vs. App 斗争不休

作者 / 张祺

2010年Wired网站曾发表文章，分析了Web在整个互联网领域的衰落原因。日前Pewinternet的一份研究报告显示，59%的受访者认为Web比单纯App更加重要和实用，但35%的参与者持相反意见，他们表示更喜欢App，相比开放式Web，App的体验更胜一筹。

05 细数25个硅谷最热创业公司

作者 / Boonsri Dickinson

硅谷被誉为创业公司的天堂，多如繁星的创业公司Karma、99Dresses、Getaround、Omada Health、SocialCam都在此处白手起家。那么，哪些创业公司值得我们重点关注？哪些是目前最火热的？本文详细介绍了硅谷当下最火热的25家，帮助读者从中捕获创业灵感、探索创业秘诀。

06 微软推出Lifebrowser

作者 / 魏兵

据ZDNet报道，微软正在研发一款名为“生活浏览器（Lifebrowser）”的新产品，它可以将使用者过去生活和工作中的一切值得记忆的里程碑事件进行分类登记，并能够通过用一个“音量控制（Volume Control）”的工具查看其中的细节。

07 微软和诺基亚陷入恶性循环

作者 / 思远

微软和诺基亚虽然都在努力推广Windows Phone平台，希望吸引开发者的关注，但由于大幅落后于竞争对手，导致它们陷入了一个难以摆脱的恶性循环。据IDC的调查，只有37%的应用开发者有意为Windows Phone编写应用，较此前的调查有所下滑，远低于iPhone的89%和Android的79%。

08 Office 15及IE 10发布时间曝光

作者 / 魏兵

曝光信息来源于微软合作商之一的Meetroo公司CEO Maarten Visser透露的一份微软2011年12月发送给合作商的路线图资料。路线图中显示Office 15将会在2013年上半年发布，而测试版可能会在2012年中期完成。IE 10将会在2012年下半年的某个时间推出。

09 Ubuntu：未来的数据中心霸主

作者 / Vaughan-Nichols

或许你认为Ubuntu只适合桌面领域，许多关于它的报道都聚焦在以消费者为中心的设计上。然而根据W3Tech的统计数据，自2011年7月起，Ubuntu在Web服务器领域已经超越RHEL，现在Ubuntu已成为企业数据中心的新生代霸主。

10 世界上最厉害的程序员都在中国

作者 / 王然

Interview Street是帮助知名科技企业招聘程序员的在线编程挑战平台。企业能在Interview Street CodeSprint平台上发布限时编程挑战，程序员可选择自己感兴趣的企业或内容参与。最近公布的一个编程竞赛结果显示，前10名中除1人国籍不明外，其他9人都来自中国。

2012年云端数据10大趋势

在过去的12个月，数以千计的云计算应用被构建和部署，《eWeek》周刊总结了有可能在2012年引领云端数据的10大趋势。

01 继续沿用现有的存储系统

对于大多数企业来说，将所有数据都存放在云端既不现实也不可能。不断扩大的数据存储需求也潜移默化地推动了对现有存储系统能力的要求。未来的一年，本地存储还会是主导。

02 私有云将在大型企业中普及

企业通常希望在公司内部使用高效和规模化的云服务以降低成本。私有云提供的可伸缩规模、灵活性和高性价比是传统基础设施解决方案无法比拟的。

03 云灾难恢复显现优势

以往企业灾难恢复不得不依靠外部专用装置实现，搭建和维护费用昂贵。而使用云服务则成为灾难恢复的新选择。虽然不可能达到零停机，但为企业提供了有力的保障。

04 企业开始选择云灾难恢复

企业SaaS应用在遇到灾难时将会导致怎样的后果？如何在企业控制范围之内创建额外的保护级别？答案就是寻找新的解决方案，在本地或其他供应商备份SaaS数据。

05 云应用更容易使用

某些业务应用完全可以迁入云中，以节省管理和维护费用。而通过强大的工具将应用程序迁移到云供应商，对于IT资源紧缺的企业不失为好的解决方案。

06 非关系数据库应对大数据浪潮

NoSQL数据库具备极强的可扩展性，能满足数百万用户所带来的TB或PB级别的数据。大数据浪潮的到来迫使许多IT企业不得不考虑用NoSQL数据库替换其传统关系数据库。

07 固态硬盘存储部署在云端

迁移到云中的应用程序并不总能保证高性能，通过提供基于SSD的高性能存储，云服务提供商将能够满足可预见的应用程序响应需求。

08 更好的数据压缩技术

云中存储每GB数据的开支仍然昂贵，重复数据删除和压缩技术能够最大限度帮助企业降低成本。在当今的技术水平下，需要发掘新技术来解决成本问题。

09 更多地使用云分析

数据分析需要可扩展的计算和存储环境，但建立一套专门的硬件分析系统非常昂贵，分析软件同样耗资不菲。而在云环境提供分析服务并采取“按使用次数付费”的模式可以为企业节约成本。

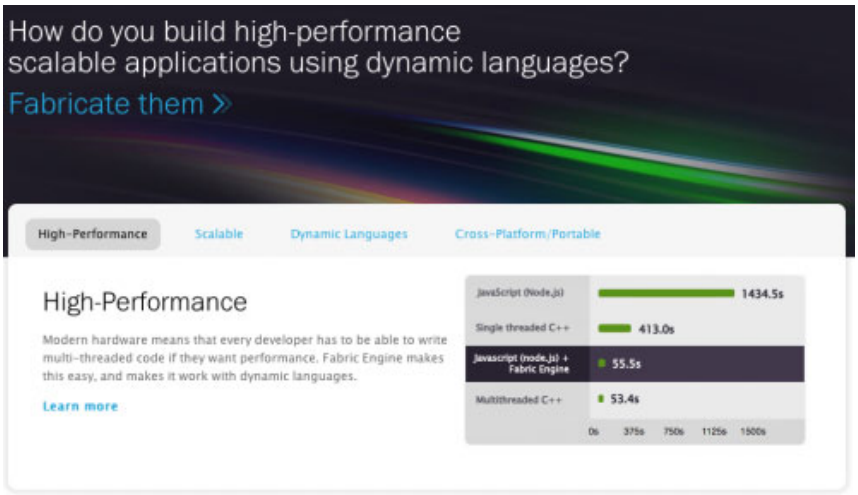
10 企业将开始追寻云计算

2012年很多公司将会选择部署云计算技术，但有的公司仍然在等待和思考。虽然企业使用云计算的模式和过程可能不同，但经济性和效率是显而易见的。

Fabric Engine 1.0

Fabric Engine是一款可以让JavaScript和Python这类的动态脚本语言完成高性能编程的解决方案，并且将陆续增加对Ruby和PHP的支持。借助Fabric Engine，现有的脚本语言开发者无需具备多线程C++开发经验，只需在程序要求高性能的部分使用特殊的类似JavaScript的KL语言编写，即可大幅度提升运行效率。

Fabric Engine可以用PyQt的方式运行在Windows、Linux和Mac OS X系统上，亦可以通过插件的方式在Firefox或Chrome中调用，还可以部署在云端服务器上。根据在Amazon EC2 Fabric Engine上使用Monte Carlo Value-at-Risk的测试结果，使用了Fabric Engine的Node.js完全可以媲美使用多线程的C++。



Qt 5.0 Alpha

跨平台GUI开发框架Qt发布5.0 Alpha版本。在这一版本中，所有平台迁移至Qt平台抽象层（QPA），QPA最早于Qt 4.8引入作为Qt Embedded的替代品，现在将应用在全部平台上。这将方便各个平台相关的代码抽象化，方便Qt向不同平台移植，比如QNX和Android。

重新构建的图形层，相比Qt 4获得了效率上的提升。Qt Quick将使用基于OpenGL的Scenegraph，依赖OpenGL ES 2.0编程；引入QOpenGL类替代QGL；从QPainter中移除平台相关的X11和CoreGraphics后端。

此外，尽管使用Qt 5的应用程序开发者看不到，但Qt 5内部也重新进行了模块化设计。此举允许不同模块之间的独立开发，方便了第三方开发者对Qt贡献代码。

GNOME 3.4 发布

GNOME 3.4是2011年4月发布3.0以来的第2个GNOME发行版。它在用户体验方面带来了大量改进，包括众多问题修正和细节改进。呈现出一个更悦目、精致、可靠的GNOME 3。

这一版本也包含了一些重要的新进步。应用程序是近期GNOME设计和开发的重点，一大批应用程序在这一发布版中得到了更新。使用者会看到构筑应用程序的开发组件方面的改进。这些改进包括平滑滚动、重新设计的用户界面元素、更精致的视觉主题和应用程序菜单（Application Menu）。

这一发布版的亮点还有：新的文档搜索功能，通过Boxes方便连接远程计算机和使用虚拟机，以及可以根据时间动态更新桌面背景。

项目管理和缺陷跟踪工具 Redmine 1.4.0 发布

Redmine是一个开源的、基于Web的项目管理和缺陷跟踪工具。它用日历和甘特图辅助对项目及进度可视化显示，同时支持多项目管理。Redmine同时也是一个开源软件项目解决方案，提供集成项目管理、问题跟踪，并提供对多个版本控制软件的支持。

新版本加入的特性包括：支持多SCM，用户可自定义多选内容，重新设计的问题复制、移动、编辑功能，解决附件更新冲突问题，提供REST API，支持Ruby 1.9和JRuby。

jQuery Mobile 1.1.0 发布

jQuery Mobile是jQuery在手机和平板设备上的版本，用于创建针对智能手机和平板电脑的跨平台Web应用。jQuery和jQuery Mobile的用户界面总设计师Todd Parker称，jQuery Mobile旨在“为jQuery社区创建一个优雅的能够兼容当前所有主流移动平台的HTML5 UI库”。jQuery Mobile不仅给主流移动平台带来jQuery核心库，而且还带来了一个完整统一的jQuery移动UI框架。

新版本对许多功能做出了改进，包括工具栏实现完全固定，页面切换更快、更流畅，新增Turn和Flow两种页面切换方式，其中Turn方式类似WP中的Metro风格，Flow方式则近似iOS中标签切换方式。此外，新版本支持多种页面切换方式，新增了淡入淡出渐变方式，并采用了全新的AJAX载入程序设计。

Notepad++ 6 发布

Notepad++是Windows平台上的一款免费文本编辑器，由中国台湾程序员侯今吾基于同是开放源代码的Scintilla文本编辑组件开发。该软件以GPL授权发布，拥有100款以上插件，具有完整的中文化接口，为众多程序员所喜爱。

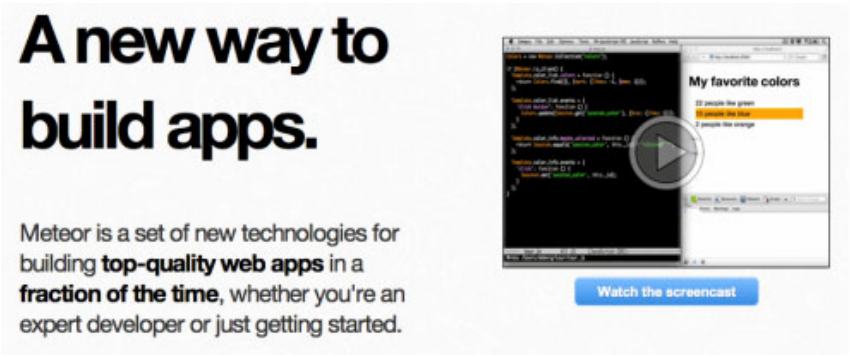
在新发布的第6版中，主要改进包括：支持 PCRE（Perl兼容正则表达式），添加了类似Sublime Text的文档概览功能，改善了大文件的加载性能。此外，Spell Checker、NppFTP、NppExport、Plugin Manager、Converter等插件也得到了更新。

全新JavaScript实时框架Meteor

Meteor是一个新出炉的现代网站开发平台，基础构架使用Node.js+MongoDB，它把这个基础构架延伸到了浏览器端，如果App用纯JavaScript写成，JS API和DB API就可以同时在服务器端和客户端无差异地调用，本地和远程数据通过DDP（Distributed Data Protocol）协议传输。

当页面CSS样式和HTML结构发生改变，可以自动刷新浏览器，实现代码的热部署，方便查看运行效果。访客浏览网站时，服务器端和每一个浏览器端的数据增删查改都将自动同步推送至服务器和每一个会话终端，不需要刷新页面来查看新内容，新版本代码和数据推送过程也不会打断当前用户的正常浏览。

即使你不在下一个新项目中使用Meteor，也可以从它的设计、实现和哲学中得到收获。



CoffeeScript 1.3 发布

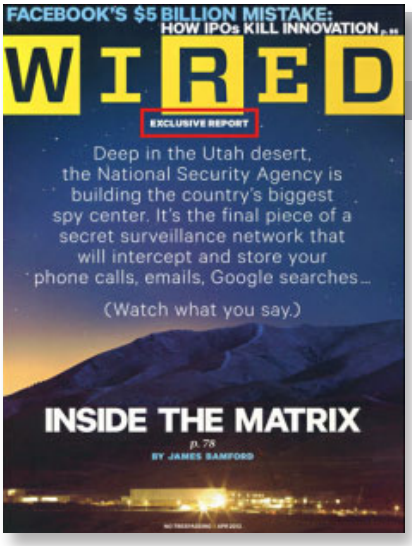
CoffeeScript是一门构建在JavaScript之上的编程语言，是Github上受关注最多的项目之一。它的语法受Ruby、Python和Haskell启发，相对JavaScript来说，代码更简短，也更优雅可读。在通常情况下，使用CoffeScript编写的代码比JavaScript简短三分之一。

CoffeeScript 1.3版包含众多更新，新版本要求代码运行在JavaScript的严格模式，REPL支持多行模式，每个编译过的文件都会生成一条CoffeeScript版本标记，类似a or= b这种形式的条件赋值语句将会被视为语法错误。

新概念IDE——Light Table

如今用的户软件界面向着越来越简洁易用的方向发展，而开发工具却似乎与此相反，恨不得让各种按钮、菜单占据屏幕上每一寸空间。有没有可能在开发工具中协调简洁的设计和强大的功能？前微软Visual Studio开发团队成员Chris Grainger设计了一款概念型的IDE——Light Table。受Bret Victor影响，Light Table与其他IDE最重要的区别是，它充分利用了交互，其中的代码是“活”的。

Light Table已经在Kickstarter上发起了集资，正式版将在2012年晚些时候发布，首先得到支持的是JavaScript和Clojure两种语言，对Python的支持也在计划当中。



Wired

2012.4

斯坦福教育实验可能永远改变高等教育

2011年秋天，位于硅谷核心地带的斯坦福大学做了一件开天辟地的事——向能够上网的人开放包括CS221（人工智能入门）在内的三门课程，教师为斯坦福大学教授Sebastian Thrun和Peter Norvig。每周，和大学课堂一样的授课内容和作业都会在线发布并自动升级；期中考试、期末考试有严格的时间要求。斯坦福大学不会向未录取的学生发放学分，但在期末时，学完课程的学生可以获得一份正式的结业证书。

这样的机会让全世界为之动容。16万学生中有三分之二来自海外，从印度到南非、从新西兰到阿塞拜疆共和国，学生遍布190个国家；100多位志愿者登记成为译者，将讲课内容翻译为44种语言；由于伊朗的YouTube遭到阻断，一位学生克隆了CS221课堂网站，经过教授们授权，开始向1000名学生重新张贴视频文件。

除了计算机专业学生，参与者还包括高中学生、人文专业学生、中学科学教师，以及七十岁的退休人员。一名学生这样描述CS221：“数字时代的在线伍德斯托克学院”。

在CS221出现的同时，另有两门计算机课程也通过其他数字平台在斯坦福大学开讲，这两门课程吸引的学生都没有超过CS221的注册人数，不过有些三门课都上的学生说CS221的教材和网站较少雕琢。两位斯坦福教授将该平台发展为Coursera——一家提供在线课程的独立企业，他们从斯坦福开始，但已扩张为其他机构为方向。他们计划在2012年开设密码学、游戏理论等14门计算机课程。目前，这些课程都是免费的。麻省理工学院宣称他们正在努力追赶斯坦福大学，他们正在创建的课程名为MITx，将于2012年秋季开设数门课程，注册、上课均将免费，不过学员如要取得结业证书，将支付“相当中肯”但仍然需要慎重考虑的费用。

CS221课程创立者Thrun教授并不担心这些令人尊敬的大学或科系挤垮他刚刚起步的事业，他憧憬着自己的数字大学。

隔墙有耳——美国国家安全局正在建设美国最大间谍中心

布拉夫代尔（Bluffdale）地处美国犹他州沃萨奇岭（Wasatch Range）东麓、奥克尔山脉（Oquirrh Mountains）西麓，为摩门教区腹心所在，是全美最大的多配偶教派之一——使徒联合教友教派（Apostolic United Brethren）的总部。与教徒们的愿望相违，如今这里正在建设的并非教堂，而是美国国家安全局（NSA）犹他州数据中心。在新建筑中装备的并非圣经，而是服务器、计算机智能专家、武装部队；这里的新来客并不聆听来自天国的福音，而是秘密采集、存储和分析流动在全球电信网络上的千奇百怪的语音和图像。

这处戒备森严的数据中心耗资20亿美元，将于2013年建成投运，占地100万平方英尺，包括访客控制中心——耗资970万美元，仅允许通过检查的人员进入；管理处——技术支持和行政人员专用场所；数据大厅——四栋各占地2.5万平方英尺的建筑，放着一排排服务器；后备发电机和燃料罐——可为中心供电至少3天；水泵房——每天可泵动170万加仑液体；制冷设备——制冷能力在6万吨左右，防止服务器过热；变电站——满足中心65兆瓦左右的电力需求；安全设施——视频监视、入侵检测和其他防范手段将耗资1000万美元左右。

犹他州数据中心一旦投运，实际上将成为NSA的云。数据将由情报机构的窃听卫星、国外窃听站、全美电信大楼中的秘密监视室提供。随后，NSA的代码破译员、数据挖掘员、分析师、反恐专家以及其他任职于总部和全国各地情报机构的工作人员将可以访问这些数据。

存储于服务器、路由器和数据库中的各种形式的信息包括私人邮件的完整内容、手机来电记录、Google搜索以及各种个人数据线索——停车收据、旅行日程、书店购物记录以及其他数字式“小零碎”。在某种程度上，它实现了第一任布什政府所创立的“全面信息意识”方案，这一方案于2003年遭到国会否决，原因是公众质疑它有可能侵犯美国公民隐私。

写给上市吸金的高科技公司

Facebook今年一旦上市，将募资至少50亿美元，它将成为全世界有史以来最大的互联网IPO。然而如此盛况却并非Facebook所乐见，而是美国证券交易委员会（SEC）按照规定逼迫的结果。按照规定，一旦一家公司的股东人数超过500，就必须登记股权，这意味着股东可以在柜台交易（OTC）市场上进行交易，公司无法控制，也无需取得公司同意或配合。没有任何一家高端企业愿意自己的股票以这种低估值的不透明方式进行交易。

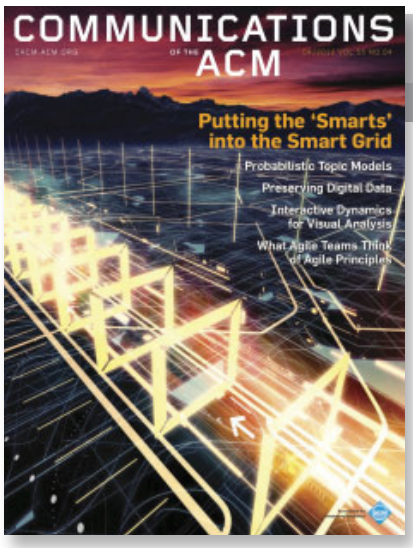
上市对于投资者和员工可能是好事，但对于公司本身却往往是坏事。上市迫使CEO们关注股市短期波动，损失的是公司的长期成长；上市破坏公司创办者对公司的控制力，将公司交到成千上万素未谋面的股东手中。对于年轻的公司来说，强行上市会导致无法持续增长的死亡螺旋。事情不必如此，科技公司还有一些更好的融资选择。

积极为科技公司提供资金的风险投资是以投资者财富增长为基础，而不是以建设成功的企业为基础，如果不想上市，创业公司可选择彻底放弃风险投资和天使投资，通过公司利润为公司提供资金。这听起来很稀奇，但仍可能取得相当大的成功。

另外就是让公司进入私有市场——即SecondMarket和SharesPost等在线平台。与公有市场不同，私有市场允许公司控制股票的购买及机密财务信息的审查。从公司的角度看来，这类市场的最大优点是局外人获得的权利以及产生的影响都不大。

美国国会正在逐步通过相关法律，SEC规则有望改变，届时科技公司就不会像Facebook那样遭遇强迫上市的命运。

科技公司需要花一些时间才能彻底探明可用的选项。可以选择的路越多，这些公司找到其他生存方式，甚至发达方式的机会就越大；公司机会越多，价值越高。硅谷的风险投资人已经接受了某些投资组合公司永远不会上市这一事实，他们甚至发明了一个专用术语——选择权价值（option value）。



Communications of the ACM

2012.4

阅读计算机科学经典著作

我们往往对于自己的特定研究领域过于投入，以至于无法充分利用核心计算机科学原本具有的潜力；我们缺乏CS（计算机科学）领域的基本理论知识；更有甚者，CS经典著作竟不为许多计算机科学家所知。

带着这些想法，土耳其伊兹密尔理工学院计算机工程系博士生导师Selma Tekir于2011年夏季学期在他所在的计算机工程系组织了几场计算机经典著作阅读会，他的团队选择了几本经典书籍和文章，启动了读书项目：C.A.R. Hoare所著《The Emperor's Old Clothes》与《An Axiomatic Basis for Computer Programming》、图灵的《Computing Machinery and Intelligence》、Ken Thompson的《Reflections on Trusting Trust》、Dijkstra的《The Humble Programmer》，以及高德纳的《计算机程序设计艺术》等。

Tekir发现，通过阅读这些经典著作，所有人都大有收获，因而决定将这样的集体阅读作为常规学术活动在伊兹密尔理工学院持续开展下去。

例如Hoare的著作将让读者了解英国1960年代以及1970年代的计算机行业，当时的编程语言社区亦在著作中详加描述。Hoare设计了一种排序算法，比Shell之前设计的效率来得更高，但因为思路独特，并不知道如何在计算机中实现。当他在ALGOL 60课程上第一次听到关于递归程序的原理时，他意识到这种机制刚好可以用来实现他设计的排序算法，这一算法后来成为QuickSort的前身。这则故事告诉我们，每个人都应该与他人交流，以此获得解决问题的方法，还应该时时想到改进前人的成果。此外，他还认为对于任何系统来说“简洁”至关重要。无论编程语言、操作系统还是任何软件都需要有一个简单、可靠的核心。基于这些概念，他发表了著名的关于公理化语义的论文《An Axiomatic Basis for Computer

Programming》，论文也同样简洁，只有短短6页，前辈们就是这样的，用6页纸就能开宗立派。

再比如高德纳，他在计算机编程方面具有远见卓识，他说："编程是由一个人告诉另一个人他想让计算机完成什么工作的艺术。"

阅读CS经典著作给读者带来永恒不变的思想，阅读CS理论知识有助于理解CS领域的局限性，开阔读者视野，让读者远离时代的喧嚣，从更朴实无华的角度评估CS领域，从而直接影响当前正在进行的研究。读者通过学习计算机历史，研究杰出科学家的工作、生活，将认识到跻身这一令人尊敬的职业领域和特权社区的真正裨益。此外，先辈轶事会鼓舞年轻学者、普及历史、凝聚社区，促使大家将CS作为一门独立学科和专业进行对待。

Selma Tekir希望他的观点能够提高读者对计算机经典著作的兴趣，由CS专家更新书目，能鼓励他们扩大经典书库范围。无论教育者还是研究者，阅读经典著作所费之时日不是浪费，而是投资。

敏捷团队眼中的敏捷原则

2001年2月，美国17位志趣相投的软件开发人员联合首创"敏捷软件开发"这一术语，确立了《敏捷软件开发宣言》和《敏捷软件十二原则》。

新的敏捷方法不断出现，旧的方法不断消亡。随着软件工程师，以及将敏捷方法视为己出的敏捷软件开发团队日益成熟、进步，敏捷宣言及其十二原则还能在多大程度上体现这些人的价值观呢？敏捷团队如何看待这十二原则呢？

带着这些疑问，研究者于2010年在美国北卡罗来纳州立大学先后进行了两次调查。

第一次调查有326人参与回答，主导问题为：下列2001年制定的敏捷原则对于2010年的敏捷团队有多重要？（1=不重要；5=至关重要，不遵守此原则就不算敏捷团队）。问题后随机列出12种原则，要求答题者进行评分。评分结果分为6等：4.6分（原则1、3）、4.5分（原则5、7、12）、4.4分（原则9）、4.3分（原则2、10）、4.1分（原则4、6、8）、3.8分（原则11）。除了评分，答题者还可以提供文字意见，研究者据此在第二次调查中对原有原则进行修订，使得修订后的原则能够凸显原有原则的行业新趋势、缺失细节或发展情况。

二次调查要求答题者对经过改进的原则给出评分和意见。第二次调查主导问题：一支团队若要成为敏捷团队，下列哪种实践可谓至关重要？（1=不重要；5=至关重要，无此实践则不算敏捷团队）。问题后列出45种实践，要求答题者进行评分。调查结果反映出回答者对增加"迭代"一词以及将"面对面沟通"改为"同步沟通"感到不满意，另外他们更喜欢言简意赅的表达。但总的来说对修订后的原则是赞同的。

最后的结论是：敏捷宣言和原有的十二原则道出了在长达十几年的时间里，与软件行业改革相伴相随的敏捷趋势的实质。

将"智能"投入电网：人工智能的大挑战

世界各地有许多政府都在采取行动向低碳经济转型，这一目标要求我们不再直接使用本已用惯的石油燃料。未来，电动汽车（EV）和高速电动火车将广为普及。相似的，家庭、办公室制暖将改用电力驱动下的地源热泵、空气源热泵供能。大部分用于满足以上需求的电力都通过再生型风能、太阳能以及潮汐能资源获得，而非依赖时下所用的煤、天然气发电，这一点至关重要。

面对这一挑战，观点日趋统一：简单拓展现有电网无法满足上述挑战，必须对电网从根本上进行重新设计。美国能源部设想了这样一种"智能电网"：全自动电力输送网络，对每一个客户和节点进行监控，确保电能和信息在电厂和电器之间、在所有节点相互之间进行双向流动。智能电网的分布式结构加上宽带通信和自动控制系统，将实现实时市场交易以及人、建筑、工厂、发电设备以及电力网络之间的无缝接口。

智能电网面临的挑战见于需求端管理、电动车辆、虚拟电厂、产消合一者涌现、自恢复网络等方面。以电动车辆为例，针对个别用户，需要根据电动车辆的日常活动和行驶需要预测电动车辆的充电需求；针对电网不同位置，需要在已知电动车辆连续行驶、电池原有电量、用户社交活动的前提下，预测该位置的所有电动车辆总充电量；需要设计分散式控制机制，协调所有电动车辆行驶（车辆的电量和充电需求各不相同），通过激励措施使消费者前往不同充电点，以此维持电网安全运作，防止变压器由于需求过大而跳闸；需要设计各种算法，优化电动车辆的充电周期，满足用户的预计需求，同时实现V2G（车辆到电网）利润最大化。

总之，面对五花八门的作用因素，智能电网技术将采用一些算法和机制逐一解决牵涉到以上因素的各种问题。

（感谢译者李芳支持）

智能眼镜——下一个杀手级便携设备？

Google很可能将Project Glass做成类似Android的开放平台，让AR技术进入之前无法进入的领域。

Google的一款新型便携电子设备近日浮出水面。Google联合创始人Sergey Brin带着Project Glass智能眼镜参加了一场公开活动，这也是Project Glass第一次出现在公众场合，比起iPhone 4的露面，Project Glass的出场方式显然高级了很多。

Google Project Glass是由Google X Lab开发的一款智能眼镜，这个实验室还有很多超越时代的项目，比如Google自动驾驶汽车。相信再过几年，Google X Lab会让Google摆脱缺乏专利的窘境。

Google Project Glass由一个水平框架、位于右侧的显示器和透明显示屏组成，采用Android操作系统。从目前公布的视频来看，Google Project Glass支持摄像头等外设，采用类似“光标”的操作方式，用户通过轻微摇晃头部来进行操作。另外，它也支持类似Siri的语音识别输入方式。同时，Project Glass支持3G/4G网络，可以连接通讯录，进行语音、视频通话。用户使用Project Glass的导航功能时，可以先使用GPS进行定位，再通过语音输入目标地址，然后增强现实可以将路标显示到眼镜上，帮助用户找到正确的目的地。而进行视频通讯时，对方可以通过摄像头看到用户目前看到的画面。

Google Project Glass无疑是一款划时代的产品，但从工程角度来看，该产品并不会取得很大的成功。Project Glass如果在近期上市，无论元器件的成熟度，还是相关产业链的成熟度都还不够，电池技术始终没有长足发展，续航时间与运算能力都会因此受到限制，是否能够提供流畅的用户体验便很成问题。开创新时代的产品命运大多如此，但它们为后来的成功者奠定了非常好的基础，并且提供了丰富的经验和教训。所以，智能眼镜有可能会成为一个便携设备的新热点，如果能改变千篇一律的外形，也许能卖到当年智能手表的数量级。

智能眼镜的技术其实由来已久，最早来自战斗机飞行员的平视显示器（HUD），使用激光镭射将各种信息投影到一块透明显示屏上，帮助飞行员锁定目标。后来HUD发展成头盔式显示器（HMD），成为美军飞行员的标准装备。

第一款民用的HMD是在1994年CES上发布的Forte VFX-1，当时还不支持增强现实功能，主要是为了解决高分辨率屏幕的显示问题。任天堂的Virtual Boy也是相似产品，只不过并非采用头戴式，受到当时技术的限制只能显示红色，且游戏不多，所以被快速淘汰了。SONY也一直活跃在这个领域中，早在1997年就推出了Glasstron显示器，而在2011年推出的3D头戴式眼镜HMZ-T1颇具未来感的设计，让所有玩家耳目一新。HMZ-T1能够利用双眼视差，呈现出3D立体场景，并且增加了环绕立体声，让用户有一种置身其中的感觉。

不过，这些HMD解决的问题是如何为用户显示虚拟世界的场景，与现实的交互并不多，我们将其称为VR头盔或VR眼镜。而Google Project Glass应该算是第一种增强现实AR眼镜，应用场景已不再局限于游戏，而会扩展到需要AR的各个领域之中。最重要的是，Google很可能将Project Glass做成类似Android的开放平台，也就是说，如果谁需要AR技术，只需要使用Project Glass和对应SDK，而不用考虑底层技术的实现，这将极大减少AR的成本，让AR技术进入之前无法进入的领域。

如果Google Project Glass推出SDK，我肯定会第一时间下载，而且还会为它学Java。

马宁

北京支点联游CTO，OpenXLive开发者。曾任微软亚洲工程院软件工程师，从事Windows Embedded Compact开发工作。加入微软之前，连续四年获得“微软最有价值专家”称号。

迁移和兼容——数据库市场的新时尚

2012年的数据库时尚变成了“迁移和兼容”，也就是各数据库厂商通过用户的手去挖竞争对手的墙角。

作为数据库市场中最具市场价值的部分——商用数据库一直牢牢被Oracle、IBM和微软占据，三者中Oracle握有市场份额优势、IBM占据技术标准优势、微软则掌握着客户端优势，三家相互制衡、相互约束；而站在外围的其他中小型开源关系数据库、Big Data数据库尽管声势不小，但从产品自身的盈利能力看，似乎非常有限。

然而“资本夜无眠”，对于数据库市场中的这些大大小小的软件公司而言更是这样，“制衡”就是要被打破的，区别于前几年各家厂商热推的“自动化管理”、“非结构化数据管理”和“数据安全管理”概念而言，2012年的数据库时尚变成了“迁移和兼容”，也就是各数据库厂商打着增强技术能力的名义，直接通过用户的手去挖竞争对手的墙角。由于数据库平台的迁移不仅涉及数据库产品的变更，同时还有数据和应用软件的迁移，因此切入点很多，另外考虑到市场地位的不同，各家推出的迁移思路呈现百花齐放的态势。

- Oracle是市场最大的在位者，它希望自己的数据库能够成为一个能量巨大的“黑洞”，不仅将Teredata、DB2、SQL Server、Sybase的用户一步步通过官方或第三方工具迁移到Oracle平台，同时以MySQL为“诱饵”，逐步将这部分原有的开源用户引导到Oracle阵营。此外，凭借对SUN的收购，Oracle进一步以“软件+硬件+经验”的方式，通过Database Machine吸引大型企业用户，力图向IBM原有小型机市场发起攻势。因为拥有Java商标，所以Oracle还可以通过影响Java技术的方向，降低其他数据库平台的Java开发者向Oracle数据库迁移的门槛。

- IBM的策略是“釜底抽薪”，直接提供一个Oracle兼容模式，就好像童年玩的游戏卡一样，面向最主要竞争产品提供一个“2 in 1”的运行环境。另外，IBM也面向SQL Server、Sybase、Teredata、MySQL、PostgreSQL等产品提供免费但功能强大、简单易用的迁移工具，并且通过SQL翻译（SQL Translator）功能，在迁移中面向异构的源数据库提供SQL方言支持。尽管受到DB2以往“阳春白雪”印象的影响，短期内DB2市场占有率相对不高，但如果Oracle不采取针对性应对措施的话，很可能导致市场天平出现倾斜。

- 微软的策略相对低调，一方面依靠低技术门槛优势，允许用户通过集成服务（SSIS）将各类异构数据库迁移到SQL Server；另一方面，在SQL Server 2012（Denali）待发布版本中增加了一个叫Partial Contained Database的功能，顾名思义就是在一个SQL Server实例（Instance）中允许Contained Database作为一个独立的单元运行。这样的数据库可以有自己的认证凭据、配置信息、字符集，而且作为一个迁移手段，Partial Contained Database与其他“新”特性最大的不同在于它不需要Contained Database采用最新的SQL Server 2012，而是可以向后兼容，从目前发布的消息看，最早可以支持到SQL Server 2005。SQL Server的兼容性就好像工作中常听到的“老人老办法、新人新办法”一样，最大程度地保证用户在既往数据库平台投入的连续性。仅从这个角度看，微软在数据库市场的主要战略似乎不如其他两家那么具有侵略性，目前迁移和兼容性设计还是以做实自己的Windows平台为主。

从迁移和兼容特性看，很多数据库“新贵”们似乎有点小气，离“兼容并蓄”的气度还有段距离，还没有在市场中找准自己的定位，什么领域似乎都想尝试。当然，这也与互联网应用的特点有关，毕竟市场很宽，还没有像企业应用市场那么“拥挤”。这些数据库“新贵”们更多考虑的是如何开拓新大陆，还没有到必须“零和博弈”直接争夺对方市场的程度。

王翔
软件架构师，主要研究方向为XML、.NET、领域设计和PKI应用。工作之余喜爱旅游、写作和烹饪。

GNU Health获得 2011年自由软件奖

GNU Health致力于加强全世界医疗专业人士合作，已被联合国大学和国际健康研究所采用。

LibrePlanet是自由软件社区中活跃成员发起的开发者会议，旨在促进自由软件的开发、推广和使用。近期在美国波士顿举办的LibrePlanet 2012大会上Richard Stallman代表自由软件基金会颁发了FSF组织评选的2011年度自由软件奖，包括自由软件促进奖和社会公益项奖两个项目。

自由软件促进奖颁发给一直为促进自由软件发展而做出贡献的业界领袖，开发者所熟悉的Python之父Guido van Rossum、Perl之父Larry Wall等传奇程序员都曾是这个奖项的得主。而今年的奖项由Ruby语言作者松本行弘（Matz）获得，这是对Matz在GNU、Ruby及其他自由软件组织近20年工作的肯定。

同期颁发的年度社会公益项目奖的候选对象是为自由软件做出贡献、且能够有效服务社会的项目或团队，历届奖项得主包括Creative Commons、Wikipedia等广为人知的自由软件项目，今年该奖项的得主是GNU Health。提供健康信息和教育服务领域自由软件的NGO组织GNU Solidario负责人，同时也是GNU Health项目的开发者Luis Falcon在会议现场接受了颁奖。

GNU Health是免费的健康和医疗信息系统，包含电子医疗病历系统（EMR）、医院信息化管理系统（HIS）和健康信息管理系统三个模块。电子医疗病历系统记录患者个人健康状况的各项数据和信息，病历数据在健康信息中用于诊断辅助，并在不同系统间实现数据互操作；医院信息化管理系统用于提高医院各项工作的效率和质量，减轻医务人员需要处理的各类事务性工作，使其集中精力为患者提供医疗服务；健康信息管理系则用于记录更多健康数据。在GNU Health中，具体包含的功能有患者预约管理、医疗器械管理、医疗费用计算、数据分析统计、医疗产品和服务管理、支付管理、医疗采购管理等不同的功能模块。

在技术实现方面，GNU Health使用Python语言开发，借助Tryton项目实现用户服务层、业务服务层和数据服务层（3-tiers）的整合，同时保证应用在模块化、可拓展性和安全性等方面具备良好的基础。此外，GNU Health整合了Python包索引（Python Package Index）以便于安装和部署，使用者只需在下载应用压缩包后执行pip install命令，即可在线装载所需的运行依赖项并执行程序。在实用功能方面，GNU Health支持CalDAV日历同步，可以将就医预约和住院治疗日历与Google日历或Apple设备进行同步；GNU Health提供了轻量级的远程过程调用协议JSON RPC支持；具备增强型的处方单据和遵从最新体系的药剂分类，包含世界卫生组织WHO提供的最新药品列表，以及增强的医疗实验室模型。作为面向全球化的健康和医疗信息系统，GNU Health还提供了最新的翻译引擎，以便于实现本地化的应用。

致力于加强全世界医疗专业人士合作，改善欠发达地区的医疗信息化现状的GNU Health项目，已被联合国大学和国际健康研究所采用作为教学和研究系统。同时项目在GNU Solidario组织的推动下，已经与尼日利亚、加纳、印度尼西亚、秘鲁及阿根廷等国的医疗组织、医生、患者建立了有效的沟通渠道，帮助其使用GNU Health项目来管理各类健康信息。

尽管获得了今年FSF颁发的社会公益项目奖，GNU Health依旧还有很多功能需要不断完善和改进。项目发起人Luis Falcon曾在邮件组中发出邀请，希望有更多志愿者参与进来，参加文档编写、本地化翻译、Bug测试、质量保证、编码开发、新功能测试等方面的工作，帮助GNU Health项目为那些难以负担软件费用的用户，提供更好的健康和医疗信息化系统。

高昂
中国标准化研究院助理研究员，从事信息技术标准化研究工作。关注开源社区，也是OSGeo中国和InfoQ中文站成员。

一个成熟的系统工程
CSGS 2012大会观感

记者 / 高松

系统工程

4月13日—14日，Digital River和CSDN在上海共同举办了第四届“走向海外——中国软件全球营销论坛（CSGS 2012）”。

会议开始，首先由Digital River MyCommerce集团副总裁艾马克和大中国区总监平嫣嫣致开场词，介绍Digital River MyCommerce在2012年的最新计划——开辟中国市场，在上海和北京组建本地化团队，邀请国外有经验的专家与大家面对面传经布道。

艾马克致开场词后，发表了《全球电子商务产品及购买销售趋势》主题演讲。他介绍，软件开发商销售渠道已从传统店面销售转到线上，更多用户通过网站下载软件。

好产品才会受到用户的喜爱，如何打造优秀的产品呢？艾马克认为，“品牌美誉度+认知度+特点=产品价值”。他在会后的采访中更具体地解释了这一话题：“很多开发者是在诸如App Store的平台上开发应用的，他们获得了很好的收入、声誉和认知度。之后他们要想在更多的平台上推广就更容易。消费者越来越多地使用网络获取信息，变得越来越聪明，所以我们也要变得更聪明、更有应对性。”

平嫣嫣谈了《如何拓展欧洲市场销售》。伴随着共享软业在北美市场竞争愈加激烈，更多共享软件业主开始进军德国、法国等欧洲国家。

如何才能拓展市场？平嫣嫣表示要端正态度、完善工作。在会后的采访中，她提出了三点建议：首先要有一些原创性的东西；其次做调研，多和国外的人员交流；最后学习更多市场推广方式，要从一开始就做品牌，不要做垃圾。

Digital River MyCommerce社群媒体经理何艾玫在演讲中谈到，一套成功营销战略的基础是建立关系以及与客户对话的渠道。在数字时代，需要借助各种公关平台增加与客户在线互动，提高公司的在线销售量。

如何利用报告衡量并改善转换率？RegNow运营经理毕杰仕认为，销量、市场趋势、客户资料、跟踪代码和市场行为都是取得成功业务的重要参考指标，可以选择自己搜集数据并编制报告。但电子商务供应商编制的报告将是非常重要的资源，它帮助企业确定当前业绩并改善发展方向，研究如何利用并阐释各种报告，更有助于企业加深对客户购买原因、方式的认识并改善流程。

Digital River MyCommerce渠道行销经理蒲丹妮的演讲主题为《龙年营销策略，重在内容》。随着新年的到来，营销策略也应有所改变。如何管理社会化内容？为了取得成功，必须随时关注产业变化，创造出令潜在客户心动的内容，还要有策略来支撑，有专员来管理用户评论，使用社交网站发展潜在用户。据她介绍，2011年移动搜索增长400%，移动已是重要趋势。我们需要给移动用户提供友好的网页，才能更好鼓励用户使用。

成熟

CSDN创始人兼董事长蒋涛的演讲主题为《中国研发“智造”大时代》。他认为走向海外，面向消费者，国外有很大的机会，也有很大的挑战。谈到软硬结合，国内是最有优势的，很多产品在深圳生产，又拥有众多优秀的开发者。在中国虽然挣钱不易，但跟深圳厂商结合起来（但是它们对软件不太懂，只注重硬件），还是有希望能把这个桥梁建起


“走向海外——全球营销高峰论坛”上多位专家分享了实战经验与心得

来的。

蒋涛希望在这个“开发者为王”的时代，为大家提供更有价值的服务，让每个人都能够创造出自身的价值。谈到国内掘金，他总结了三点：第一，投身移动领域，中国智能手机已成井喷之势；第二，与电子商务的结合，手机随身携带就等于每天都在做交易；第三，开放平台，国外公司这两年能够飞速发展，跟云计算有直接关系。

万兴软件董事兼高级副总裁傅宇权发表了《从差异到价值》主题演讲，从三个方面介绍价值的体现：1. 根据营销战略的规划路径；2. 产品在差异化战略基础上的DVD规划思路；3. 价值传递的核心要素。

福昕软件项目实施经理王航在演讲中介绍，福昕PDF软件，主要销售区域在美国，其次是在亚洲、欧洲，它还讲述了福昕的发展历程。

无锡阿达游软件有限公司董事长顾方认为创业路上有一点需要时刻牢记——反思。为了做成一件事情需要细化做很多事，虽然看起来是正确的，但如果时机不对或者次序不对，就可能导致结果出现偏差，怎样在对的时候做对的事情，怎样回顾与反思走偏的路，应该能找到一种方法，而不是依靠感觉。

金山办公软件CTO葛珂演讲主题为《办公软件发展趋势》。谈到了办公软件的三个转折点：一是1988年DOS操作系统下的文字处理系统；二是1995年Windows操作系统下的文字、表格和演示套软；三是基于云计算和开放平台技术的移动办公。办公软件也从桌面向网络与在线办公转移，使得办公更高效，软件使用率更高。

海豚浏览器联合创始人刘铁锋介绍了《移动互联网海外掘金之路》，特别针对iOS、Android和Windows Phone三大平台，从发布方式、应用类型、案例分析、推广盈利等环节，系统分享海豚浏览器对移动互联网海外市场的认识和探索经验。

争鸣

在论坛环节，CSDN总编刘江提出“近年共享软件行业发生了什么变化？”就此话题，嘉宾发表了各自的观点。

傅宇权认为变化有两个，第一是传统软件逐步走向移动互联网；第二是游戏也是软件，在做游戏的同时，万兴不会放弃过去的业务。

曦力软件副总裁马笑峰认为，从2008年金融危机开始，多媒体行业软件或多或少受到了影响，2012年多媒体市场开始呈萎缩的状态，但仍有上升空间。

结语

在此次盛会中，来自国内外的演讲嘉宾就全球营销策略、具体实施经验技巧及成功案例做了精彩分享。相信未来会有越来越多的国内开发者，在摸索和借鉴这些经验的同时，带领自己的产品成功地走向海外。

Cloud Foundry：云时代的 Linux/LAMP

记者 / 杨爽

VMware软件研发副总裁Mark Lucovsky在大会上激情演讲

2012年3月28日和30日Cloud Foundry开发者大会先后在北京和上海两地召开，两千多名开发者到现场与全球第一个开源PaaS平台亲密接触。

会议当天，VMware多位资深专家，包括公司软件研发副总裁Mark Lucovsky、开发者关系高级总监Patrick Chanezon、中国云应用平台及服务总经理任道远和知名Java专家Chris Richardson等在会议上作了精彩演讲，阐述Cloud Foundry（官方网站：http://www.cloudfoundry.com）的远大愿景，并全面介绍了Cloud Foundry实践与应用的各种技术细节。

众所周知，PaaS可以使应用的部署、运行和横向扩展变得更容易。PaaS平台能够使开发人员只需专注于编程，而无需操心操作系统、虚拟机、中间件、数据库甚至运维和架构等诸多烦心事宜。但在Cloud Foundry之前，各PaaS基本上都是封闭体系，构建在专有平台之上，提供的能力也有诸多限制。有的只支持有限的语言和框架，有的不提供云应用所需的关键服务，应用程序部署范围也有各种限制。

而Cloud Foundry横空出世，迅速成为业界焦点。相对于其他PaaS平台，Cloud Foundry的优势不言而喻：系出VMware这样的名门，由大名鼎鼎的技术权威Lucovsky领军，不仅支持从桌面微云、私有云到公共云等多云环境，还广泛支持Java、Ruby、Node.js、Scala等多种语言，Spring、Rails、Sinatra、Grails和Lift等框架，以及MySQL、PostgreSQL、MongoDB、Redis、VoltDB、RabbitMQ等各种服务。

更重要的是VMware开放的心态：首先项目技术是开源的，采用Apache 2.0许可证发布，代码托管在GitHub；而平台方面，在VMware自己运营的CloudFoundry.com之外，第三方不仅可以用Cloud Foundry架设私有云，还可以架设公共云平台，支持更多特性。公共云平台appfog基于Cloud Foundry并增加了PHP语言支持，AppState建立起支持Perl与Python的私有云，Tier 3则为Cloud Foundry社区提供.NET支持。

VMware大力支持合作伙伴，推动社区，而Cloud Foundry社区的迅速建立和发展，则进一步巩固了这一PaaS平台的竞争优势。

更加值得一提的是，本次大会上中国PaaS平台新浪SAE已经宣布基于Cloud Foundry增加Ruby支持，盛大云负责人Tuoc Luong也表示将加入Cloud Foundry社区。

2012年4月11日，Cloud Foundry迎来第一个生日。值此之际，VMware公开了Cloud Foundry中部署工具BOSH和IaaS支持接口CPI的代码，实现了产品的完全开源；宣布要加强对开源社区代码贡献者的支持，公布了新的代码签入审核流程和工具；同时声明，虚拟化IaaS层面即将支持Amazon AWS和vCloud，开源社区可通过实现CPI接口，支持更多类型的IaaS。

全球越来越多的开发和部署工具开发商、公共云运营商、私有云提供商、应用开发商正在加入这一开源大家庭，任道远在演讲中称Cloud Foundry要成为“云时代的Linux/LAMP”，这一目标看来已经不是遥远的梦想。

探究中国化的云计算落地之路

记者 / 董世晓

云计算和移动互联网作为当前IT业界的两大焦点，一直为大家所津津乐道。在国内，与移动互联网的有声有色相比，云计算虽热，但总给人一种只打雷不下雨的感觉，最明显的就是缺乏能够深刻影响到最终用户的实施和应用。因为软件公司和开发人员关心的是云计算如何帮助他们切实地赚到钱，所以我们需要的是真正的云落地。

中美云落地之异

美国的云计算产业看起来已经理顺，步入正轨。作为业界普遍认可的云计算三大标杆企业，Amazon、Google、Salesforce分别提供IaaS、PaaS、SaaS这三种云计算服务的形态。

IaaS作为云计算的基础平台，PaaS、SaaS都在其基础之上构建。在美国，有众多的云计算创业公司基于Amazon提供服务，例如提供面向个人存储服务的Dropbox，估值已达到几十亿美元；提供面向企业存储和应用的，也可以支持第三方开发的软件在上面运行；Heroku、Engine Yard基于Amazon为其他软件公司的开发人员提供Pass服务，而Salesforce收购Heroku的花费竟达到令人乍舌的两亿多美元。在Google、Microsoft等的PaaS平台之上，有众多的企业级软件公司都在转做SaaS，在那之前，针对每家客户都要单独开发，而实际上如果按行业来区分的话，同一行业内是有很多共性需求的，无非把代码稍作修改，就能达到客户需求。而SaaS就能够实现将行业共性提取出来，基于Web提供服务，既为客户节省了资源，又加快了对需求变更的响应。

这些机遇和形势，国人都看到了，但好像总是行动不起来，云计算的各个环节，看起来都不是那么尽如人意。

诚然，美国的IT产业环境与我们很不相同：Amazon作为一家互联网巨头，拥有雄厚的技术实力和沉淀，能够引领美国的云计算发展；美国云计算的发展比较不至于受到各方面因素的阻碍。最根本的，在CSDN总编刘江看来，国内没有Amazon这样一个IaaS平台。如果连基础的IaaS这一层都没有做好的话，在它上面的PaaS、SaaS也就无从谈起了。虽然有几家大的互联网公司已经开始提供类似于Amazon的服务，但均刚刚起步，还存在各种各样的问题。此外，国内整个产业环境也是一大桎梏——山寨成风，使得大家互相之间不是那么信任。

那么，在中国，谁能扮演Amazon的角色，提供IaaS服务？

运营商？是的！

在2012年3月30日举行的Power应用开发商新闻发布会上，IBM系统与科技部大中华区Power Systems服务器产品总监李红对记者表示，国内能够扮演Amazon角色提供IaaS服务的只能是运营商。

李红，1991年毕业于北京航空航天大学，先在北京航天制造技术研究院担任了三年软件工程师，这期间，由于所开发的软件涉及众多的跨平台移植，让她得以精通VAX、Apollo、RS/6000等多个平台。1994年加入IBM，那时距IBM中国公司正式成立仅两年。加入IBM两周之后，RS/6000部门正式成立，李红成为这个部门最早的员工。在IBM的这18年时间里，李红经历了RS/6000以来的全产品系列，对与云计算密切相关的服务器市场，看得很深入。在李红看来，经过这些年的发展，IBM的产品已慢慢摆脱了高高在上的形象，转变得更为亲民，并且其品质在众多的关键业务应用中得到了验证，IBM的Power Cloud架构势必会在中国的云计算发展中发挥重要作用。

李红认为，中国有自己的国情：理念虽好但不一定可以实施，因为这往往不是技术问题，而是一种商业模式或者社会文化——东西握在自己手中的话，心里踏实、易于操作。此外，云计算在中国会有很多限制，存在比较大的地域差异。

互联网企业虽有野心成为中国云计算的引领者，但整体动作还是偏慢，而近几年运营商的语音、短信等的业务增长明显放缓，特别是来自微信、FaceTime等的冲击，它们迫切需要寻找新的业绩增长点；同时，运营商原有的数据中心比较简单，提供的基本都是物理方面的低附加值的主机租赁和带宽服务，所以它们也希望向高附加值的方向发展。

这时候，作为IT（信息技术）和CT（通信技术）和有机融合的ICT（信息通信技术），成为一个不错的选择。ICT业务是面向集团客户的个性化需求，对网络通信服务、信息内容应用、信息技术产品进行必要的整合和开发，形成完整的应用解决方案，并将该解决方案应用到集团客户实际生产管理过程中的“一站式”服务信息产品的拓展。电信业正走向全业务运营模式，而依托高可用数据中心的“云计算”正在改变ICT业务的实现方式。

运营商推动中国云计算，存在多方面的优势：已有的数据中心基础，为它们提供了便利；在数据的安全性方面，由于对国有企业的思维惯性，即使发生什么问题，也会有保障，所以一般客户都会信任它们；一定的政策优势。

而运营商在IT方面并不具有优势，因此，它们希望与IT厂商合作。作为提升各行业应用的最佳基础架构平台，Power拥有性能优势、绿色节能技术、PowerVM虚拟化技术、高可靠性和安全性以及强大的支撑体系，可以为ICT业务提供高效的Power Cloud平台，因此Power是电信运营商ICT业务的理想合作伙伴，可以为应用开发商实现新成长创造更多商机。

刘江认为，Amazon模式现在成为一种主流，好像国人也就知道这种模式，而实际上运营商可能更会主导云计算落地。Power架构类似于企业级的苹果，从硬件到软件，都是自己做，整个系统架构强调软硬件一体设计，因此更利于优化。因此，云计算使得Power有机会从原来的关键业务应用拓展到更广泛的范围。

云计算需要榜样案例

“榜样的力量是无穷的”。因此，我们迫切需要云计算的成功案例来发挥示范引领作用，推动云计算在中国的落地。这些案例必须符合云计算的定义，而且确实为客户带来了效益。

从目前来看，天子星的餐饮业应用和富基融通的零售业应用是两家比较有特点的代表。

天子星是一家餐饮软件公司，提供从餐饮信息化管理咨询到信息化管理系统开发、实施、培训、售后，从软件服务到硬件选型、测试的餐饮业整体IT解决方案。面对餐饮连锁企业IT系统运营管理方面所遇到的投入大、风险高、运营维护团队建设周期长等种种挑战，天子星采用Power710/730应用服务器成功构建了电信公有云运营模式，让数以万计的中小型餐饮连锁企业无需再建门店数据中心，就可以使用“天子星餐饮连锁RIF管理系统”，大幅

李红：IBM正在全力推进云计算在中国的落地

节省了成本。天子星副总裁于朝斌表示：“天子星与IBM合作开发了国内唯一采用SOA架构、基于RIF平台、实现跨平台实时通信技术的大型餐饮连锁信息管理系统。面对新一代信息技术与工业、服务业的深度融合与创新，我们构建了Power电信公有云，首次实践了‘中国餐饮信息化管理云服务’的新模式。”

富基融通是一家提供消费品和零售业软件和服务的公司。为帮助零售连锁企业彻底解决散布各门店IT系统带来的管理烦恼，实现总部机房集中管理，富基融通基于Power710/730应用服务器，构建了Power企业私有云。富基融通市场和战略合作伙伴总监王小健表示：“基于Power Cloud新一代企业级基础云平台和IBM技术团队的大力支持，我们成功地改变了以往x86的分散部署模式，建立了领先的Power私有云，从系统性能、占用空间、功耗、电力总成本各方面都取得了非常好的成效。Power Cloud能够快速响应客户的业务需求变更，更好地推动客户的业务创新。”

可以说，天子星模式是将已有的行业解决方案云化，以云的方式交付和提供，从零开始购买硬软件、开发全新应用的公有云。与之相比，富基融通的企业私有云部署模式，是通过对原有IT设施的云化管理，平滑地将现有应用迁移到云平台。

在李红看来，IT经历了“虚拟机—C/S架构—胖终端—瘦终端—虚拟机”的过程，实现了螺旋式上升。因此，云计算的发展道路同样是曲折的，但前途是光明的。

梁捷
UC（优视科技）技术总裁。1998年毕业于华南理工大学计算机专业，后长期耕耘于中国的电信和互联网市场，在电信及网络计算领域拥有超过10年的技术研发和管理经验。

梁捷专栏

基于云端架构提升移动阅读体验

兼谈UC小说阅读器

移动阅读时代到来

阅读一直是人们对个人电子设备的基础需求。在当下这个互联网时代，越来越多的作品借助网络渠道发行，上网看书成为一种潮流。

手机作为便携的上网设备，更是让移动阅读的习惯盛行开来。等车、等电梯、等她（尤其是逛街的时候）……个人移动电子设备具有很强的便携性，碎片的时间被有效利用起来。人们已经进入了移动阅读的时代。

但在手机上进行阅读，会遇到很多新的体验问题，本文就带各位读者看看UC浏览器是怎样解决这些问题的。

在小屏幕上的阅读难题

在小屏幕上快速找到想看的小说一直是人们在电子设备上阅读的主要内容之一，特别是小说连载网站，一直以来访问排行都很靠前，读者很想第一时间读到更新的章节。但只有少数小说站点推出了供手机用户阅读的页面。相信很多读者跟我有类似的体验，使用手机浏览器打开小说站点的Web页面后，很难迅速在小小屏幕上面找到想要阅读的内容。

我们就开始思考，怎样用UC独特的“云端架构”去帮助解决这个问题。

我们首先是注意到有不少用户喜欢泡在论坛里。于是做了一个独特的功能，叫做“论坛模式”，利用UC浏览器的服务器内容分析引擎，将论坛页面中的主体文字内容提取出来，然后重新进行排版。这样一来，用户访问论坛时可以看到一个非常清爽的版面，可以更有效率地进行阅读，获取关注的信息。

论坛模式可以将广告剔除了手机屏幕，更加突出了版面的主体内容。用户可以更容易找到想要查看的内容。同时论坛模式能够帮用户节省不少流量，让浏览速度快了很多。论坛模式推出之后，受到了很多用户的喜爱。

搞定难缠的图片小说

解决了一般性的排版问题之后，棘手的问题又来了。

一个用户打电话来，说她在看到收费的VIP章节时遇到一个问题：出于版权保护的原因，网站上的小说不是文字形式呈现，而是以图片形式呈现。这个图片由很多的小字组成的。她在手机上面看的时候，那些字会非常小，很难看得清楚。如果放大了之后，图片的宽度又太宽了，需要左右的滑动屏幕，非常吃力。

针对这个新情况，我们的技术团队进行了深入的研究。最终确定了一个在UC排版服务器上，进行图象处理的解决思路。

这个方案的原理是，当服务器识别出小说站点的图片页面之后，会将图片按文字进行切割，切出一个个小字。然后根据手机屏幕大小

阅读图片小说

等因素，再重新拼凑在一起，形成一个能够在手机上面方便阅读的小说图片。

这个功能同样是在服务器上进行自动适配的，用户无需进行任何设置。我们内部把这个功能称为“图片小说模式”。有了这个功能，UC的用户就能方便地读到最新的收费章节了。这个功能发布之后受到了很多用户的追捧。

阅读体验再升级：小说阅读器

随着阅读需求的增加，很多开发商在智能手机上推出了专用的小说阅读软件。它们最大的优点是，交互体验好、操作简单、内容纯粹、大多具有很好的书签功能。

但这些软件都是需要下载安装才能使用，不同软件中的小说内容也各不相同，并没有像在Web站点上的全面。

于是我们就在想，是否能使用最新的Web App技术，通过HTML5+CSS+JS的方法，来实现类似专用小说阅读软件的功能呢?

经过技术团队的讨论，我们认为这个想法是完全可行的。经过两个星期的开发，我们做出了一个功能和客户端应用完全一样的Web App。

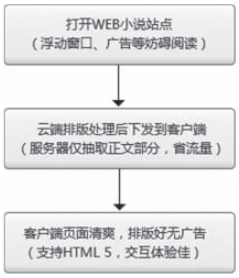
小说阅读器

这个应用不需安装，只要用UC浏览器访问这个Web App就可以实现精美的书架、字体、纸张、翻页效果等功能，手机上的阅读体验又再提升了一个层次。

而且这个Web App有良好的跨平台性，可以在Android、iOS等多个平台上使用。这样高的开发效率，在做本地应用（Native App）开发时，是完全不可想象的，本地应用单是针对Android平台兼容性做的工作往往就要搞一个月以上。

在我们的构思中，未来只要遇到可以识别的小说网站，用户就能自动切换到“小说阅读器”，这样一来，UC浏览器就具有了自动适配内容的操作形态，成为名副其实的“变形金刚”。

云端架构，使UC浏览器更“聪明”



小说阅读器架构

其实从阅读模式到小说模式，再到现在小说阅读器，我们使用的技术架构一直没有改变，就是基于UC浏览器的云端架构进行创新。当用户正常打开页面时，UC的服务器就可以针对不同类型的网站进行进行智能识别，并为用户选择最佳的展现模式。

用户只要安装了UC浏览器就不需要额外安装其他阅读软件。展示问题都可交由UC服务器智能内容识别引擎来完成。不单是页面清爽，还最大限度提高了浏览速度，节省了上网流量。

UC现在已经做了将近8年，是一家产品技术为导向的公司。我们希望能够充分地发挥在移动互联网技术领域积累的优势，通过我们在产品技术上的创新，为整个行业创造价值，帮助用户享受移动互联网时代的生活。

对话：迅雷下一代交互界面引擎 Bolt

记者 / 卢鹄翔

针对当前交互界面开发的热点和解决方法，迅雷首席工程师刘智聪和CSDN总编刘江进行了对话，就开发迅雷下一代交互界面引擎Bolt过程中的经历，并对Bolt的关键技术进行了阐述。

刘江：你怎么理解交互设计与软件产品开发这个领域？

刘智聪：从过去仅以实现软件功能为目标，到后来为用户提供一致的操作习惯，交互设计在现代软件产品开发中，特别是互联网产品开发中占有越来越重要的位置。如今开发者还渴望交互体验能基于用户的使用心理进行设计，在美观的同时能够提高产品的整体品味。目前市值最高的苹果公司，也正是凭借交互设计出色的软件和工业设计完美的硬件，赢得了广大消费者的青睐。

大多数开发者编写的第一个图形应用程序就已经使用了一些交互开发技术，然而软件产品的交互开发从来都不是令人愉悦的事情。由于IDE工具的强大，许多开发者容易总结出交互开发就是“拖拖控件、改改属性、写写响应”的经验，容易被认为是低技术含量的工作。但实际上，这是一个特别不容易的工作：因为作为软件产品的脸面，上至公司老板，下至普通用户，大家都可以对工作成果品头论足，这些修改意见反映到产品的方案修改上，与项目复杂度之间并不呈线性关系。很多刚从事这一行的项目经理常常不能理解，为什么按一个方案修改交互只需1天，而另一个看起来似乎差不多的方案却要改上1个月。

刘江：交互开发技术的变迁经历了哪些阶段？

刘智聪：产品的基础交互体验水平，通常是由产品所在的系统平台决定的，如果希望产品的交互体验水平能超越平台框架所提供的基础体验，就需要各种界面库、控件库、引擎库，这个现象在Windows平台上尤为常见。纵观整个IT行业，其他领域的优秀开源库层出不穷，却少有被公认和广泛使用的界面库。界面库的历史如下。

第一代：SDK开发，使用系统默认控件。

第二代：基于窗口子类化的自绘控件皮肤库，绘制通常基于GDI，控件类型和系统一致。

第三代：提供一套完整的体系（窗口、绘制等），所有的控件都基于这个体系开发，提供很多功能更强的控件，并有统一的方法开发和使用新控件。由于Windows系统限制，第二代库有很多功能局限，第三代库主要是解决实现能力问题。

第四代：布局文件+脚本语言的开发模式，依靠“组合”抽象界面并围绕这个概念搭建。不提供控件但提供易用的控件开发模式，不提供内置特效但能让使用者开发自己的界面特效。刘智聪将迅雷Bolt定义为第四代界面库。

刘江：Bolt界面引擎是怎么诞生的？

刘智聪：作为一款流行的下载软件，迅雷5.8在完成了功能部分的提升后，希望进一步改进产品的交互体验，使其更美观、更现代。迅雷6是基于传统Windows界面开发技术改进交互体验的结果，虽然呈现出的效果不错，却带来了另一个严重的问题——开发成本大幅度提高。作为一款由多个部门合作开发的客户端产品，传统的开发技术此时已不能很好地组合迅雷各个部门开发的模块，开发成本与稳定性都存在问题。公司迫切需要一个能解决这些问题的界面引擎。于是在2009年，Bolt被提上了日程。

Bolt界面引擎开发团队通过观察交互设计师的工作流程，提出了“完全基于原子对象组合交互”的

概念，强制在开发中使用MVC模式。界面引擎基于这个理念，提出“引擎不提供控件，但提供开发控件”这种“授人以渔”的方法，希望通过让开发者掌握新的思路来提高生产力。

与前一个版本相比，采用了Bolt界面引擎的迅雷7，在保证原有功能并重新设计了全部界面的基础上，代码量减少了至少40%，平均每个需求的修改成本也由原来的0.8小时降低到了0.3小时。在交互性、可维护性上也都有了大幅提升。迅雷7技术总负责人表示“Bolt界面引擎让开发效率提高了1倍以上”。

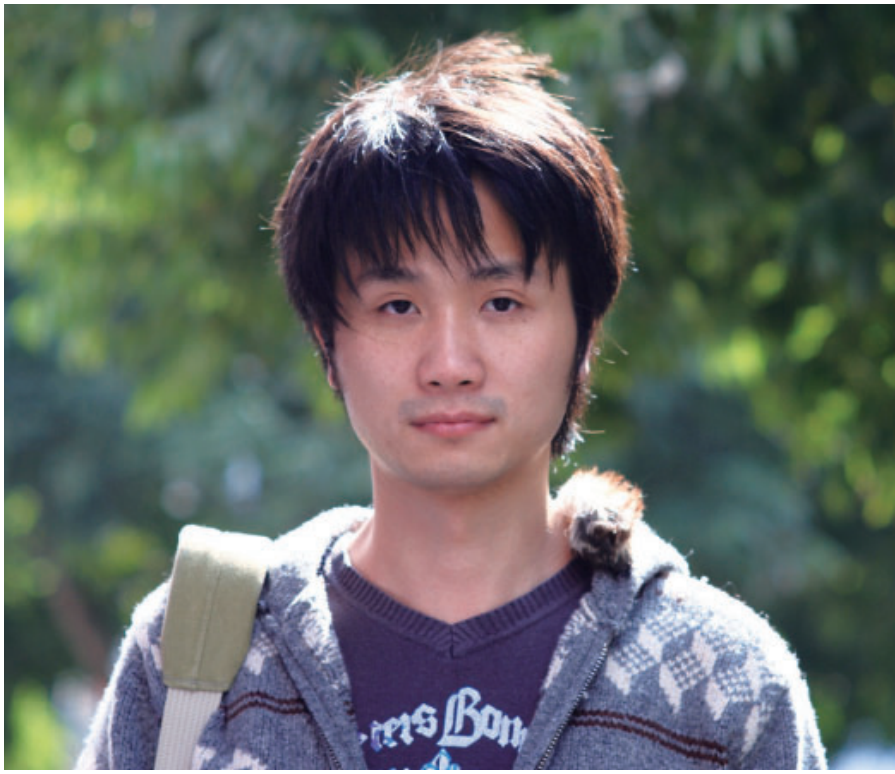
刘江：Bolt界面引擎关键概念和应用情况如何？

刘智聪：Bolt界面引擎的核心理念可以用一句话描述：通过定义原子UI对象（UIObject）之间的父子关系和位置关系组成对象树（UIObjTree）来描述界面，按逻辑构建的UIObjTree实际上组织了绘制函数里“按什么样的顺序在什么地方画什么”的问题。实现上，有两个关键概念——HostWnd和Render。Render可以把构建完成的UIObjTree渲染成一张位图。而HostWnd是界面引擎核心与操作系统之间的桥梁，能把这张位图通过系统提供的API画到屏幕上，并能转化操作系统的键盘、鼠标等事件成为引擎定义的标准输入事件。

DirtyRect（脏矩形）是驱动Render工作的核心。当一颗UIObjTree产生了脏矩形，其对应的Render在下次渲染时就会开始有动作，否则就什么也不干。Render会选取与脏矩形相交的所有UIObject，然后按这些UIObject的z-order排序，再按这个顺序依次调用UIObject的Draw方法。Draw的实现会调用优化过的图形图像绘制函数，这一切就构成了Bolt的高速渲染引擎。

Bolt界面引擎的概念，并不依赖于操作系统。因此，它的发展方向之一是移植到各个平台，目前最成熟稳定的是Windows，已有多款迅雷产品应用了这项技术，在Android和Mac上也有初步移植的版本。

不过要想让界面引擎在移动设备上流畅运行，原有的基于CPU指令集优化的高速渲染器还不能满足要求。首先CPU性能不足，其次耗电量高，因而迫切需要使用设备提供的硬件加速功能，目前迅雷正在尝试多种支持硬件加速的框架方案。



刘智聪介绍了迅雷Bolt的诞生过程和关键技术

刘江：Bolt引擎适合哪些开发者使用？

刘智聪：开发者需要首先掌握一门简单高效的脚本语言Lua，还需要学会使用XML。Bolt目前只专注于交互开发，仅掌握它并不能完成一个完整的应用产品，还需要掌握使用C/C++给Lua环境扩展功能的方法。Bolt界面引擎是为了能高效开发需要长期维护与改进的工业级产品而设计的，并不是一种能快速上手，拖动控件便可完成产品的快速RAD开发框架。但总的来说，Bolt界面引擎的学习曲线平缓，根据迅雷的经验，对于有交互界面经验的开发者，一周左右的学习和实验就可以基本掌握。

Bolt界面引擎的开放刚刚走出了第一步，重点是分享迅雷过去在交互开发上的经验，介绍迅雷发现的新思路。Bolt提供3种免费授权方式：非商业授权、小型商业授权和大型商业使用授权。

本期封面报道，有来自美国硅谷云计算公司的一线报道，有典型互联网技术发展历程的精辟总结，有视频网站YouTube累积七年的可扩展经验，更有来自微软、盛大、新浪、淘宝、百度、腾讯、有道、爱奇艺、中科院的众位嘉宾披挂上阵，从IaaS、PaaS、SaaS三个层面全方位分享云计算技术一线的成果。

美国云计算印象

文 / 蒋涛，刘江

3月19日至3月26日，我们参加中国电子学会云计算专家委员会组织的“云计算美国之行”访问团，到西雅图和旧金山两地拜访美国多家云计算相关公司，其中既有Amazon、微软、Google、Facebook、Rackspace这样的行业巨头，也有Heroku、Engine Yard、Puppet等新兴云计算公司，还包括CloudCamp为代表的美国云计算社区。虽然行色匆匆，难以进行非常深入的交流，但接触面比较广，对美国云计算产业和技术的发展情况有了更多直观认识。

微软

我们到访的第一站是位于西雅图的微软总部。企业IT走向云计算已成为大势所趋，所以微软与软硬件巨头IBM、HP、Oracle、Dell一样，都在积极布局云计算技术、产品和平台，或收购，或积极研发。

为什么现在云计算如此重要？负责微软企业云战略的Rolf Harms从经济学角度给出了答案：

- Cloud = doing what you do today for less
- Cloud = doing what you do today better
- Cloud = doing things you can't do today

微软自己的官方网站Microsoft.com迁移到Azure平台后，成本降低了90%，可用性从99.1%提高到99.997%，部署时间却从数周下降到45分钟，改善巨大。

Harms的介绍中有一幅图讲述私有云和公共云的不同使用场景，非常清晰，如图1所示。

而长远看，公共云相比私有云具有明显的价格优势，如图2所示。

云计算大战的焦点之一，是如何降低数据中心的建设和运营成本以及环保问题。Facebook开源Open Compute项目更加剧了数据中心技术方面的竞争。微软也大幅投资云计算数据中心，发展新一代数据中心。这次我们有幸参观了微软在Redmond Ridge的研发数据中心，各方面都非常先进，虽然设备的密度极大（52U/rack），PUE值仍然能够达到1.17。应该说，在纯粹设备上国内并不落后，但这个数千平米的数据中心只有10人负责运营，而支撑Azure平台的主力Quincy数据中心，占地47万平方英尺，运营人员也只有35人，自动化程度惊人。另外负责人介绍，由于设备密度太大，思科现有的交换机已经无法支持，必须开

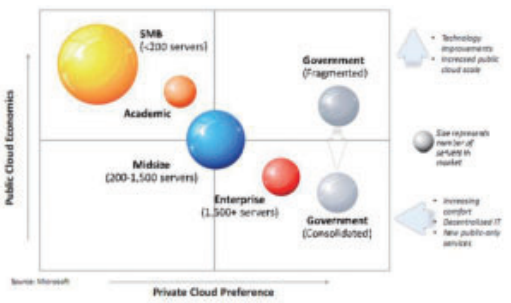


图1　私有云与公共云适用场景

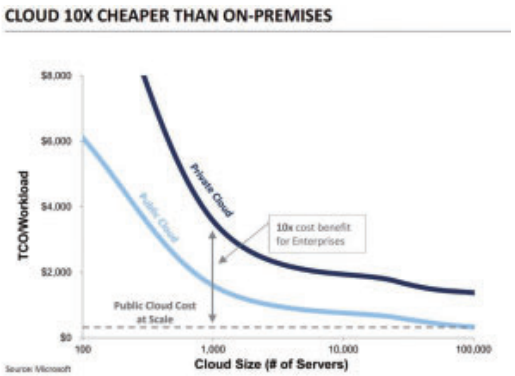


图2　私有云与公共云成本比较

发新一代。

Amazon

云计算的理念其实已经发展了很多年，而作为一个术语而流行始自2006年，但至今仍然免不了众说纷纭，这一点国内外情况倒是差不多。云管理平台企业enStraus副总裁Bernard Golden在美国CIO圈子里颇有名气，他在交流中说到一个很有意思的现象：很多大公司的CIO虽然一方面口口声声云计算还不安全，不能轻率采用，可另一方面已经非常普遍地使用Salesforce、Workday等SaaS软件管理客户和员工等最敏感的信息。不过他也承认，虽然在美国云计算已经成为主流，但大家还是免不了不断的分歧与纷争，因为有太多背景不同的厂商和人说自己在做云计算。

Source: Gartner (December 2011)

图3 Gartner的IaaS魔力象限

但有一点是大家都有共识的，那就是Amazon已经在公共云领域比较稳固地建立了自己的优势。2011年12月Gartner的研究表明，AWS自2006年推出以来，一直保持高速的产品研发节奏。AWS云平台无论在执行能力还是在前瞻性上，都在IaaS提供商中处于领先地位。而且它也在向PaaS扩张。

与此对应的是，AWS惊人的指数型发展速度。由图5可见，截止到2011年底，Amazon S3已存储了7620亿对象，年增长率达到192%，而且还在不断加速。另外，据美国调查公司451Group的报告，AWS已经占据了美国59%的IaaS市场份额。

虽然Amazon很少公布图5那样具体的规模数字，但我们还是可以从各种渠道做一些估算。此前Accenture的Huan Liu曾估算AWS的服务器可能有44.5万台，但一般都认为这个数字可能偏高。James Hamilton去年6月曾透露，AWS当时每天增加的容量足可以支持Amazon最初头五年全球运营，而那时公司营收已经接近30亿美元了。另据4月中旬DeepField的数据，现在每天有1/3的美国互联网用户至少访问一次AWS，它还占据了1%总互联网流量，并已经是全球第四大CDN。

拜访Amazon云计算部门时得知，它现在约有2000多人（其中有相当比例是销售人员），今年还准备扩招500人左右。有熟悉美国企业运作的同行者据此粗略算出，Amazon今年的营收目标至少应该是20亿美元。而2011年Gartner对此数字的估计是10亿。发展速度的确惊人。

云计算最初的用户主要是新兴公司，但现在大型

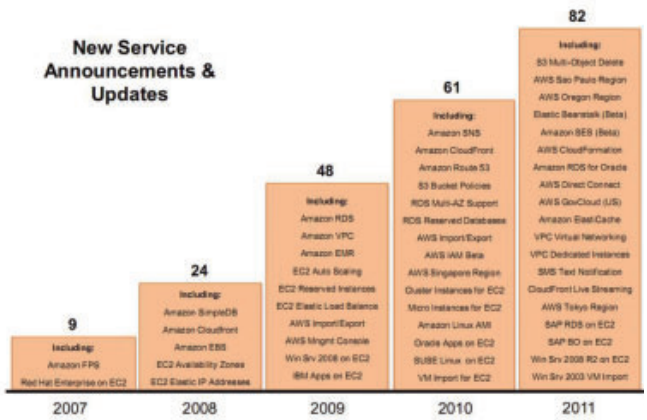


图4 AWS历年发布的服务（出自Jeff Barr的演示稿）

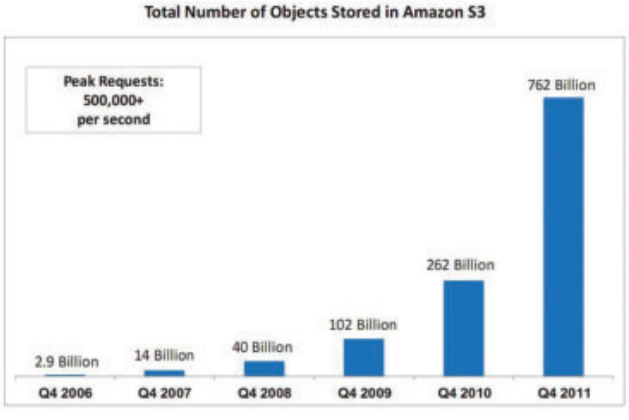


图5 Amazon S3所存对象数的增长（出自Jeff Barr的演示稿）

互联网企业和许多大型公司也已经转向AWS。其中视频服务商Netfilix是一个经典案例，这个流量巨大的网站完全放弃了自建数据中心。利用AWS做海量存储、大数据分析和容灾备份的就更多了：SAP、Adobe、三星、爱立信、日立、趋势……

值得注意的是，除了Web应用和海量数据领域，AWS近年也在开拓面向企业云服务的产品，IBM、Microsoft、Oracle、SAP的全线商业产品和各开源产品的都可部署在AWS云服务上。4月底，Amazon又推出应用商店AWS Marketplace，由Amazon负责统一计费，更加方便用户。

Rackspace与OpenStack

此次我们也访问了Rackspace在旧金山的办公室。Rackspace是韩裔美国人Richard Yoo于1998年创办的，最初只是一家普通的ISP。在传统的主机托管厂商之中，较早转型云计算，也成为IaaS领域另一个领导性厂商，市值接近80亿美元，2011年营收超过10亿美元。最近他们透露，总服务器数已经超过8万台，云计算方面已经占到总业务的两成以上，据估计，它的规模是Amazon云平台的五分之一。

Rackspace在云计算领域的领导地位还有一个重要原因——它是目前关注度最高的开源云计算项目之一OpenStack的主要发起者和实际领导者。而OpenStack的核心开发人员有些就在旧金山办公室。到访当天他们显得很忙，未及深谈，后来得知，不久后第五个OpenStack版本Essex发布了。

从技术上说，OpenStack其实与其他开源云技术项目相比并不具备绝对优势。但由于拥抱开源早，路线图明确，很快获得了急需IaaS和私有云平台技术的众多厂商的支持。很大程度上，OpenStack已经成为抗衡Amazon霸主地位的阵营中坚。当然，社区运营和生态系统建设方面，OpenStack也有很多成功经验可供国内企业借鉴。现在参与开发的公司除了Rackspace、Nebula之外，还有Red Hat、Nicira、HP、Canonical、DreamHost、新浪、维基百科、思科甚至Citrix。

在美国的交流中，我们发现除了大家言必称

图6　与Rackspace创业者关系负责人Robert Scoble合影

OpenStack之外，VMware主导的开源PaaS项目Cloud Foundry的关注度也较高。有趣的是，两者都将自己的目标定为成为云时代的Linux。相比之下，Citrix最近捐献给Apache的CloudStack（主要创始人是当年Sun JVM的作者Sheng Liang）和老牌的Eucalyptus的人气就差了很多，但它们都与AWS兼容，可以列入Amazon阵营。这三大生态系统之间如何竞合，将是云计算的大看点之一。

云计算推动创业

比AWS等云平台本身发展更重要的是，它们已成为美国云计算和创业生态系统的基石。由于云平台的存在，加上移动互联网的重大机遇，大量创业公司得以迅速兴起，硅谷、纽约、波士顿等各地，到处一派繁荣景象。比如在旧金山，我们到访的一座不起眼的四层小楼里，《Wired》杂志、Wikia和Engine Yard都在其中。

AWS的客户名单（数以十万计）中包括许多美国当前热门公司（其他多是广告平台）：Pinterest、Dropbox、Instagram、Reddit、Zynga……

这些新兴企业的网络服务就架设在AWS平台上，从而免除了“为了让灯亮起来就要在IT上花费80%的时间和成本”（Gartner语），可以专注于开发应用、满足用户需求。比如，最近刚以10亿美元（因为大部分是股票，实际价值可能更高）被Facebook收购的Instgram，其技术方案大量采用AWS（主机选择Amazon EC2，图片数据库采用Amazon S3，CDN选用Amazon CloudFront等）。所以虽然Instgram只有13名员工（工程团队仅3

truste.com	20.7	vindicosuite.com	3.6
invitemedia.com	18.4	adnetik.com	3.6
chartbeat.net	16.2	tapjoyads.com	3.4
evidon.com	14.6	rubiconproject.com	3.4
adsafemedia.com	11.7	redditmedia.com	2.9
kenshoo.com	10.9	tidaltv.com	2.5
andomedia.com	9.5	lucidmedia.com	2.5
netflix.com	9.1	optimizely.com	2.5
pinterest.com	7.5	spotxchange.com	2.2
sharethis.com	7.3	heroku.com	2.2
tubemogel.com	6.9	adsrvr.org	2.0
addthis.com	6.8	adroll.com	2.0
dropbox.com	6.3	adometry.com	1.9
instagram.com	4.6	datalogix.com	1.9
echoenabled.com	4.6	zynga.com	1.8
rapleaf.com	4.3	mediaforge.com	1.8
ooyala.com	4.3	permuto.com	1.7
brightroll.com	3.9	verticalacuity.com	1.6
bizo.com	3.8	admarvel.com	1.6
liverail.com	3.7	imgur.com	1.5

图7 AWS上每天访问量最大的客户（出自DeepField的演示稿）

人），却构建了最强大的移动端图片分享平台，甚至让Facebook感到了威胁。

除了面向消费者的互联网和移动应用之外，云计算也为技术提供商创造了机遇。本次访问中我们在CloudCenter巧遇Puppet创始人Luke Kanies，并进行了简短的交流。他是一位资深的系统管理员，后转开发，2003年用没学多久的Ruby写出Puppet，是DevOps的最初推动力之一。后以此辞职创业，因为采用开源模式，最初三年公司基本没有收入，是苦苦撑下来的。今天，Puppet已经用于Google、Twitter、Zynga等数千企业，2011年底累计融资2500万美元。Puppet的劲敌，是Amazon的灾难处理大师Jesse Robbins开发的Chef，但他本人现在只担任Opscode的首席社区官。而Rackspace旧金山办公室的负责人Alex Polvi也是系统管理员出身的成功创业代表，他的云管理和监控服务CloudKick去年被Rackspace收购。

值得注意的是，Amazon现在已经并不限于Infrastructure这一层。从图8中还可以很清楚地看到，它也在同时向Platform层发展，加入很多管理和监控服务。这毫不奇怪，IaaS公认利润不高（虽然比Amazon的老本行还是要高一些），PaaS和SaaS才是肥沃之地。而且从用户的角度来看，一站式的服务永远是有吸引力的。

但基于AWS的各种第三方PaaS服务仍然在繁荣发展，我们这次访问的Engine Yard和Heroku都是其中的优秀代表。

Engine Yard是历史最久规、模最大的PaaS平台，

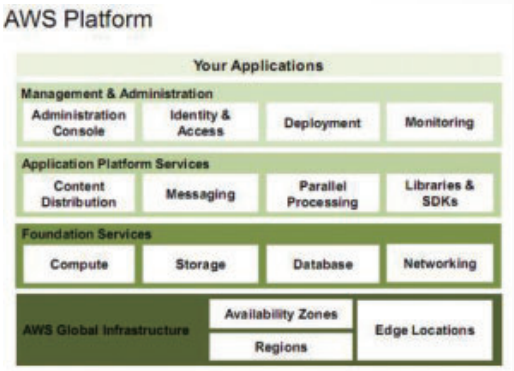


图8 AWS其实已经不只是IaaS

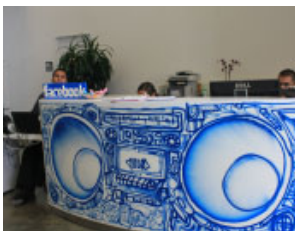
有Amazon投资。目前付费用户超过2400个，包括Apple iTunes在欧洲的平台也依靠它们支撑。目前90%的应用为Ruby，其次也支持PHP和Node.js。

而Heroku则是一家以用户体验和技术精湛而著称的PaaS公司，最初主要支持Ruby（Ruby创始人Matz是其首席架构师），现在也覆盖了Python、Java、Scala和Node.js。2010年以2.12亿美元卖给Salesforce，是云计算平台领域最大的并购之一。但有意思的是，收购已经过去一年多，Heroku仍然独立经营，只是从最初的地址搬到了旁边一座更大的楼里，完全保持创业公司的样子，而且平台也丝毫没有要从AWS迁移到自家Salesforce基础设施的打算。同样有趣的是，Facebook不是自建平台，而是在2011年9月与Heroku合作，让后者为Facebook上的社会化应用开发者提供PaaS服务。为了这次合作，Heroku特别增加了PHP支持。而合作也为Heroku带来丰厚回报：用户激增，平台上的应用数超过140万。

有竞争，但合作共赢是主流，大家把更多精力放在产品和创新，为用户提供更好体验上。这种和谐共生的现象，尤其令我们这些见惯了乱景的中国人印象深刻。

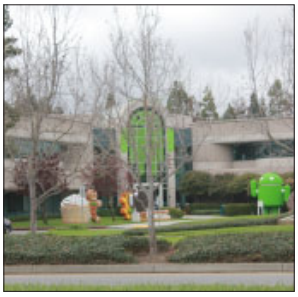
今天，一个普通的技术人员可以短时间内借助云计算平台，拥有和巨人对手们相同的计算资源，实现梦想，这才是云计算的真正价值所在。我们需要共同为之努力。

代表团合影



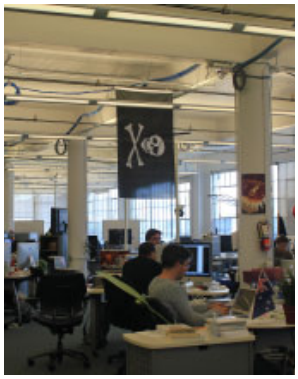
Facebook前台，难怪有人说这里像高中

到处都是涂鸦，还有游戏机，Facebook不是像高中，更像幼儿园

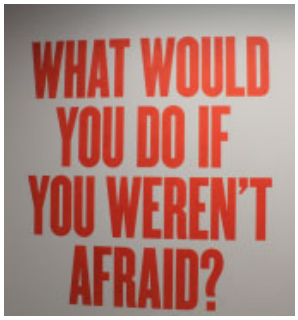


Google总部，Android大本营。据说Andy Rubin极力将团队人数控制在200以内，一座楼就能放下

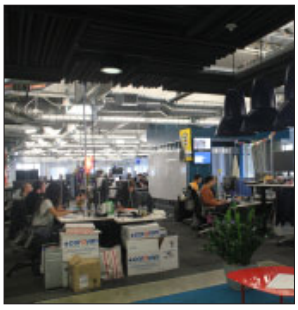
这里诞生了2亿多美元的公司——Heroku



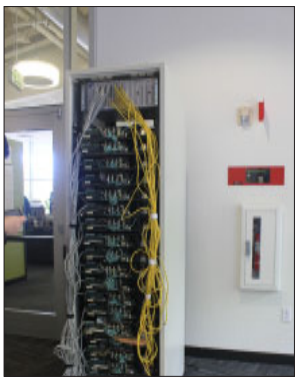
Engine Yard办公室一角，Hacker=海盗？



Facebook墙上的这条箴言，似乎未外传过，啥意思呢？

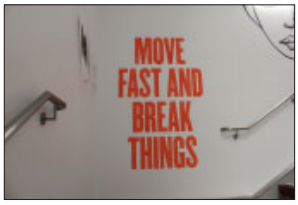


Facebook办公室一角

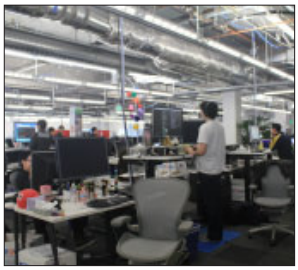


Google早期的机柜，都是四处捡来的廉价硬件，看来云计算得这样穷搞才有戏

Heroku墙上的项目进度仪表盘很棒，开发方来自西班牙

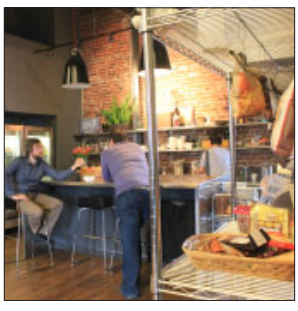


Facebook办公楼过道墙上除了涂鸦之外，就是这些现在很著名的箴言

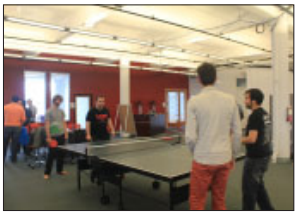


站着编程也很时髦，摄于Facebook

闻名遐迩的Google食堂，只不过和中国的问题一样，人太多了



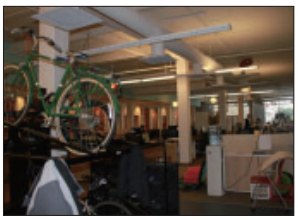
Heroku办公室一角，温馨吧？



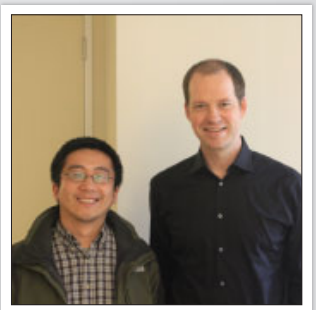
Engine Yard员工在打乒乓球，还有正式比赛，看来是旧金山创业公司最流行的运动，这咱们中国人擅长啊

Facebook的编程语言墙，什么都有，我猜是各位同学的处女语言

Facebook的新办公楼很气派，原来是Sun公司的总部，所谓“千古兴亡多少事，不尽长江滚滚流”啊



旧金山创业员工多住在城里，自行车成了主要交通工具



Puppet创始人Luke

摄影/文字：刘江

可伸缩性的10年探索

知名网站的技术发展历程

文 / 林昊

互联网已经发展多年，其中不乏脱颖而出者，这些网站多数都已存在了接近10年或10年以上，在如此长时间的发展过程中，除了业务上面临的挑战，在技术上也面临了很多的挑战。

在本文中，我挑选了一些Alexa排名较前的网站（网址为http://www.alexa.com/topsites，排名截止到2012年4月21日），看看它们在技术上是如何应对业务发展过程中的挑战的。

Google目前Alexa排名第1。它诞生于1997年，当时是一个研究性项目，每个月build一次索引，build出来的索引通过sharding（shard by doc）的方式分散到多台服务器（Index Server）上，具体的网页数据同样通过sharding的方式分散到多台服务器（Doc Server）上，当用户提交请求时，通过前端的一台服务器将请求提交给Index Server获得打了分的倒排索引，然后从Doc Server提取具体的网页信息（例如网页标题、搜索关键词匹配的片段信息等），最终展现给用户。

随着索引的网页增加，这个结构可通过增加Index Server以及Doc Server来存储索引以及网页的数据，但仍然会面临其他很多方面的问题，于是在这之后的十多年的时间里，Google做了很多事情来改进上面的结构。

1999年，Google增加了一个Cache Cluster，用来Cache查询的索引结果和文档片段信息，同时将Index Server和Doc Server通过Replicate的方式变成了Cluster。这两个改造带来的好处是网站的响应速度、可支撑的访问量以及可用性（Availability）得到了提升。

上面的这个变化造成了成本的增加，Google在硬件方面的风格始终是不用昂贵的高端硬件，而是在软件层面来保证系统的可靠性及高性能，于是同年，Google开始采用自行设计的服务器来降低成本。

2000年，Google开始自行设计DataCenter，采用了各种方法（例如采用其他的制冷方法来替代空调）来优化PUE（能源利用率），同时对自行设计的服务器也做了很多优化。

2001年，Google对Index的格式进行了修改，将所有的Index放入内存， 这次改造带来的好处是网站的响应速度以及可支撑的访问量得到了极大的提升。

2003年，Google发表了文章Google Cluster Architecture，其Cluster结构组成为硬件LB+Index Cluster+Doc Cluster+大量廉价服务器（例如IDE硬盘、性价比高的CPU等），通过并行处理+sharding来保证在降低对硬件要求的同时，响应速度仍然很快。同年Google发表了关于Google文件系统的论文（GFS在2000年就已经上线），这篇论文很大程度也体现了Google不用昂贵硬件的风格，通过GFS+大量廉价的服务器即可存储大量的数据。

2004年，Google再次对Index的格式进行了修改，使得网站的响应速度继续提升。同年Google发表关于MapReduce的论文，通过MapReduce+大量廉价的服务器即可快速完成以前要使用昂贵小型机、中型机甚至是大型机才能完成的计算任务，而这显然对于Google快速地构建索引提供了很大的帮助。

2006年，Google发表了关于BigTable的论文（2003年开始上线），使得海量数据的分析能够达到在线系统的要求了，这对于Google提升网站的响应速度起到了很大的帮助。

以上3篇论文彻底改变了业界对于海量数据的存储、分析和检索的方法（小道消息：Google内部已完成了GFS、MapReduce、BigTable的替换），也奠定了Google在业界的技术领导地位。

在一些场景中，Google也采用MySQL来存储数据。同样，Google对MySQL也做了很多修改，它使用的MySQL信息可以从https://code.google.com/p/google-mysql/了解。

2007年，Google将build索引的时间缩短到分钟级，当新网页出现后，几分钟后即可在Google搜索到，同时将Index Cluster通过Protocol Buffers对外提供Service，以供Google各种搜索（例如网页、图片、新闻、书籍等）使用，除了Index Cluster提供的Service外，还有很多其他的Service，例如广告、词法检查等。Google的一次搜索大概需要调用内部50个以上的Service，Service主要用C++或Java来编写。

2009年，Google的一篇《How Google uses Linux》文章，揭示了Google在提升机器利用率方面也做了很多的努力，例如将不同资源消耗类型的应用部署在同一台机器上。

在之后，Google又研发了Colossus（下一代类GFS文件系统）、Spanner（下一代类BigTable海量存储和计算架构）、实时搜索（基于Colossus实现），主要都是为了提升搜索的实时性以及存储更多数据。

除了在海量数据相关技术上的革新外，Google也不断对业界的传统技术进行创新，例如提高TCP的初始拥塞窗口值、改进HTTP的SPDY协议、新的图片格式WebP等。

在Google的发展过程中，其技术的改造主要围绕在可伸缩性、性能、成本和可用性4个方面，Google不采用昂贵硬件的风格以及领先其他网站的数据量决定了其技术改造基本都是对传统的软硬件技术的革新。

facebook

Facebook目前Alexa排名第2。它采用LAMP构建，随着业务的发展，它也在技术上做了很多改造。

作为改造的第一步，Facebook首先在LAMP结构中增加了Memcached，用来缓存各种数据，从而大幅度提升系统的响应时间以及可支撑的访问量，之后又增加了Services层，将News Feed、Search等较通用的功能作为Service提供给前端的PHP系统使用，前端的系统通过Thrift访问这些Service。Facebook采用了多种语言来编写各种不同的Service，主要是针对不同的场景选择合适的语言，例如C++、Java、Erlang。

大量使用Memcached以及访问量的不断上涨，导致访问Memcached的网络流量太大，交换机无法支撑，Facebook通过改造采用UDP的方式来访问Memcached，以降低单连接上的网络流量。除此之外，还有其他一些改造，具体信息可以查看http://on.fb.me/8R0C。

PHP作为脚本语言，优势是开发简单、易上手，劣势是需要消耗较多的CPU和内存。当Facebook的访问量增长到了一定规模后，这个劣势就比较突出了，于是从2007年起，Facebook就尝试多种方法来解决这个问题，最后诞生于Facebook Hackathon的HipHop产品成功地脱颖而出。HipHop可以自动将PHP转化为C++代码，Facebook在使用HipHop后，同等配置的机器，可支撑的请求量是之前的6倍，CPU的使用率平均下降了50%，从而为Facebook节省了大量主机。将来Facebook还会对HipHop进行再次改进，通过HipHop将PHP编译为bytecode，放入HipHop VM中执行，再由HipHop VM来编译为机器代码，方式与JIT类似。

2009年，Facebook研发了BigPipe，借助此系统，Facebook成功让网站的速度提升了两倍。随着Facebook访问量的上涨，收集众多服务器上的执行日志也开始面临挑战，于是Facebook研发了Scribe来解决此问题。

对于存储在MySQL中的数据，Facebook采用垂直拆分库和水平拆分表的方式来支撑不断增长的数据量。

作为Facebook技术体系中重要的一环，Facebook也对MySQL进行了很多优化和改进，例如Online Schema Change等，更多信息可见http://www.facebook.com/MySQLAtFacebook。

发展之初的Facebook采用了高端的存储设备（例如NetApp、Akamai）来存图片，随着图片不断增加，成本也大幅提高，于是2009年Facebook开发了Haystack来存储图片。Haystack可采用廉价的PC Server进行存储，大幅度降低了成本。

Facebook除了使用MySQL存储数据外，近几年也开始摸索采用新的方式。在2008年Facebook开发了Cassandra，在Message Inbox Search中作为新的存储方式。不过在2010年，Facebook又放弃了Cassandra，转为采用HBase作为其Messages的存储，并在2011年将HBase应用在了Facebook更多的项目上（例如Puma、ODS）。据说，现在Facebook更是在尝试将其用户以及关系数据从MySQL迁移到HBase。

从2009年开始，Facebook尝试自行设计DataCenter以及服务器，以降低其运行成本，并对外开放了其构建的PUE仅1.07的DataCenter的相关技术。

Facebook在技术方面的基本原则是："在能用开源产品的情况下就用开源，根据情况对其进行优化并反馈给社区"。从Facebook的技术发展历程上可以看到这个原则贯彻始终，Facebook的技术改造也主要是围绕在可伸缩、性能、成本和可用性4个方面。

Twitter目前Alexa排名第8。在2006年诞生之时是采用Ruby On Rails+ MySQL构建的，2007年增加了Memcached作为Cache层，以提升响应速度。

基于Ruby on Rails让Twitter享受到了快速的开发能力，但随着访问量的增长，其对CPU和内存的消耗也让Twitter痛苦不堪，于是Twitter做了不少改造和努力，例如编写了一个优化版的Ruby GC。

2008年Twitter决定逐步往Java迁移，选择了Scala作为主力的开发语言（理由是"难以向一屋子的Ruby程序员推销Java"），采用Thrift作为其主要的通信框架，开发了Finagle作为其Service Framework，可将后端各种功能暴露为Service提供给前端系统使用，使得前端系统无需关心各种不同的通信协议（例如对于使用者可以用同样的调用服务的方式去访问Memcache、Redis、Thrift服务端），开发了Kestrel作为其消息中间件（替代之前用Ruby写的Starling）。

Twitter的数据存储一直采用MySQL，发展过程中出现的小插曲是，当Facebook开源了Cassandra时，Twitter本计划使用，但最终还是放弃，仍然保持了使用MySQL，Twitter的MySQL版本已开源（https://github.com/twitter/mysql）。Twitter也是采用分库分表的方式来支撑大数据量，使用Memcached来Cache tweet，timeline的信息则迁移为用Redis来Cache。

2010年，Twitter在盐湖城拥有了第一个自建的DataCenter，主要是为了增加可控性。

从Twitter的发展过程看，6年来它的技术改造主要围绕可伸缩以及可用性。

作为一家电子商务网站的员工，请允许我在此介绍这个Alexa排名21的著名电子商务网站的技术演变。

1995年，eBay诞生，当时采用CGI编写，数据库采用的是GDBM，最多只能支撑5万件在线商品。

1997年，eBay将操作系统从FreeBSD迁移到Windows NT，另外将数据库从GDBM迁移为Oracle。

1999年，eBay将前端系统改造为Cluster（之前只有一台主机），采用Resonate作为负载均衡，后端的Oracle机器升级为Sun E1000小型机，同年给数据库增加了一台机器作为备库，提升可用性。前端机器随着访问量不断增加还可以应付，但数据库机器在1999年11月时已经达到了瓶颈（已经不能再加CPU和内存了），于是在11月开始将数据库按业务拆分为多个库。

2001-2002年，eBay将数据表进行了水平拆分，例如按类目存储商品，同时部署Oracle的小型机换为Sun A3500。

2002年，将整个网站迁移为用Java构建，在这个阶段，做了DAL框架来屏蔽数据库分库分表带来的影响，同时还设计了一个开发框架以供开发人员更好地上手进行功能开发。

从eBay的整个发展过程来看，技术改造主要围绕在可伸缩性和可用性两点。

Tencent 腾讯

腾讯目前Alexa排名第9。最初QQ IM采用的是单台接入服务器来处理用户的登录和状态保持，但在发展到一百万用户同时在线时，这台服务器已经无法支撑。

于是QQ IM将所有单台服务器改造为了集群，并增加了状态同步服务器，由其完成集群内状态的同步，用户的信息存储在MySQL中，做了分库分表，好友关系存储在自行实现的文件存储中。

为了提升进程间通信的效率，腾讯自行实现了用户态IPC。之后腾讯将状态同步服务器也改造为同步集群，以支撑越来越多的在线用户。

在经历了前面几次改造后，已基本能支撑千万级别的用户同时在线，但可用性比较差，于是腾讯对QQ IM再次进行改造，实现了同城跨IDC的容灾，加强了监控和运维系统的建设。

此后腾讯决定对QQ IM架构完全重写（大概是2009年持续到现在），主要是为了增强灵活性、支持跨城市的IDC、支撑千万级的好友。在这次大的技术改造过程中，腾讯的数据都不再存储于MySQL中，而是全部存储在了自己设计的系统里。

从QQ IM的技术演变来看，其技术改造主要是围绕在可伸缩性和可用性上。

淘宝网

2003年，淘宝诞生，直接购买了一个商业的phpAuction的软件，在此基础上改造产生了淘宝。

2004年，将系统由PHP迁移到Java，MySQL迁移为Oracle（小型机、高端存储设备），应用服务器采用了WebLogic。

2005-2007年的发展过程中，用JBoss替代了WebLogic，对数据库进行了分库，基于BDB做了分布式缓存，自行开发了分布式文件系统TFS以支持小文件的存储，并建设了自己的CDN。

2007-2009年对应用系统进行垂直拆分，拆分后的系统都以Service的方式对外提供功能，对数据采用了垂直和水平拆分。

在进行了数据的垂直和水平拆分后，Oracle产生的成本越来越高，于是在之后的几年，淘宝又开始将数据逐渐从Oracle迁移到MySQL，同时开始尝试新型的数据存储方案，例如采用HBase来支撑历史交易订单的存储和检索等。

近几年淘宝开始进行Linux内核、JVM、Nginx等软件的修改定制工作，同时也自行设计了低能耗服务器，同时在软硬件上进行优化，以更好地降低成本。

从淘宝的整个发展过程来看，技术改造主要围绕在可伸缩性和可用性两点，现在也开始逐渐将精力投入在了性能和成本上。目前淘宝的Alexa排名为第14。

总结

从上面这些Alexa排名靠前网站的技术发展过程来看，每家网站由于其所承担的业务不同、团队人员组成不同、做事风格相异，在技术的不同发展阶段中会采用不同的方法来支撑业务的发展，但基本都会围绕在可伸缩性、可用性、性能以及成本这4点上，在发展到比较大规模后，各网站在技术结构上有了很多的相似点，并且这些结构还将继续进行演变。

林昊

目前就职于淘宝，2007-2010年负责设计和实现淘宝的服务框架，此服务框架在淘宝大面积使用，每天承担了150亿+的请求；2011年开始负责HBase在淘宝的落地，目前淘宝已有20个以上的在线项目在使用HBase。

YouTube成功秘诀

七年可扩展经验分享

文 / Todd Hoff　　译 / 陈向伟

开始想做一个交友网站，最后却做成了一个每天有40亿访问量的视频共享网站（YouTube），这一路下来一定会收获颇多。最早参与YouTube开发的工程师Mike Solomon在PyCon大会上发表了题为《Scalability at YouTube》的精彩演讲。

这场演讲的关键信息就是用一些非常简单的工具就能做很多事情。在很多团队纷纷迁移到更加复杂的生态系统时，YouTube却在力求简单，主要用Python编程，用MySQL作数据库，坚持使用Apache。即便规模这么大了，YouTube实现新功能也是从非常简单的Python程序开始。

这并不意味着YouTube做的东西不够棒，事实上它们很棒。做事需要的是思路和方法，而不是什么技术把戏。那么YouTube成为全球最大视频网站的秘诀是什么呢？阅读本文便可知晓。

统计数据

- 每天40亿次的访问量
- 每分钟上传60小时的视频
- 超过3.5亿台设备访问YouTube
- 2010年营收增长一倍
- 视频数量上升9个数量级，但开发人员只上升2个数量级
- 100万行Python代码

软件架构

Python：YouTube的大多数代码都是用Python编写的。实际上，用户每次观看YouTube视频就是在执行一系列Python代码。

Apache：你想摆脱它，但却办不到。Apache是YouTube里最受欢迎的技术，因为它让YouTube保持简单。每个请求都要通过Apache。

Linux：Linux的好处在于，总有方法查看系统表现。不管应用表现多么糟糕，都可以用strace和tcpdump等Linux工具查看。

MySQL：被用得最多。用户观看YouTube视频就是在从MySQL获得数据。有时将MySQL用作关系数据库或Blob类型存储，其实都是在对如何组织数据进行权衡和决策。

Vitess：它是YouTube发布的新项目，是用Go语言编写的MySQL前端。Vitess做了很多动态优化，重写了查询，能充当代理的角色。它基于RPC，目前能处理所有YouTube数据库请求。

Zookeeper：分布式锁服务器，用于配置信息的维护，是一种非常有趣的技术。但使用起来比较难，所以需要阅读使用手册。

Wiseguy：CGI servlet容器。

Spitfire：模板系统。它有一个抽象语法树做数据转换，从而使YouTube的速度更快。

序列化格式：不管使用哪种序列化格式，其代价都非常昂贵，所以需要仔细权衡。不要使用pickle，它会导致协议缓冲区缓慢。YouTube有自己编写的BSON实现，要比其他序列化格式快10~15倍。

常规经验

YouTube的原则：选择最简单的解决方案，因为需要灵活地解决各种问题。而一旦具体指定某些东西，就会限制住自己的手脚。

整个过程与可扩展性相关：可扩展的系统不会成为你的绊脚石，这点你可能没有意识到。它不是空谈，而是一种通用的解决问题的思路。

大系统设计的特点：每个系统都会根据特定的需求量体裁衣。一切取决于正在构造的具体内容。

YouTube不是异步的：一切都是阻塞的。

思辨，不教条：保持简单。什么意思呢？当你遇到了，你就会明白。如果你做代码审查时修改了成千行代码和很多文件，那就意味着或许有更简单的方式去做这件事情。第一个Demo应该保持简单，然后迭代。

解决问题就一个词：简单。寻找最简单的方法来解决问题。复杂的问题有很多，但第一个解决方案不需要复杂。随着时间推移，复杂度自然会提高。

很多YouTube系统都是从单个Python文件开始的：多年后才逐渐成为大型生态系统。这些系统的原型都是用Python写的，它们存活的时间让人惊叹。

在设计审查中：首个解决方案是什么？打算如何迭代？是否知道将来怎样使用这个数据？

事情会随着时间而改变：刚建立的YouTube跟以后的YouTube毫无关系。一开始它只是个交友网站。如果一直把它作为一个交友网站来设计的话，一定没有今天的YouTube了。因此要保持灵活。

YouTube的CDN：最早它是被承包出去的。后来由于费用太高，YouTube决定自己做。如果有好的硬件配置，就能构建出相当出色的视频CDN。建一个大机架，然后把机器放进去，接着安装lighttpd，覆写404处理程序找到那些之前找不到的视频。整个过程花费了两个星期并且第一天就承受了60Gb的数据流量。用简单的工具就能做很多事情。

必须权衡：Vitess替换了协议族中的HTTP实现。因为即使用C实现，运行依然缓慢，所以YouTube去掉了HTTP，用Python直接进行Socket调用，使全局的CPU成本降低了8%。HTTP的封装是非常昂贵的。

可扩展性技术

这些都不是新思路，但令人惊讶的是几个核心思路竟然可以应用到很多方面。

分而治之：扩展性技术

扩展性技术的一切都和工作划分有关，决定该如何去执行。它适用于很多地方，从Web层开始，你有许多Web服务器。这些服务器既相互联系又相互独立，而且可以进行水平扩展。这就是分而治之。

数据库分片的关键在于，如何划分数据，并让已划分的数据相互关联。从一开始就要搞清楚这个问题，因为它将影响到以后如何扩展。

简单而松散的耦合非常重要。

Python的动态特性在这里体现了它的优势。不管API有多糟糕，都可以通过存根、修改或装饰等方式解决许多问题。

近似正确：小小的欺骗

另一个不错的技术。要相信监控报告的系统状态。如果用户不能确定系统出现偏移或不一致，那就是没有问题。

真实示例。如果你在写一篇评论，而其他人同时在加载该页面，那么他们在300~400毫秒内看不见你写的东西。因为读取页面的用户并不会在乎这个评论，而评论作者在乎，所以要确保作者能看见自己所写的东西。这里要进行小小的欺骗。此时，系统不必全局一致，因为那样做代价很高而且没有必要。不是每条评论都关乎财务交易，所以可以适时进行欺骗。

构建不同的一致性模型

你了解自己的一致性模型吗？评论最终会不会一致？租一部电影的情况完全不同，因为租电影时涉及到钱，所以我们会竭尽全力不丢失任何一致性。不同的一致性模型需要取决于数据。

将熵加回系统

热词一直都在YouTube的群组中。如果你的系统没有抖动，那将面临惊群（thundering herds）。分布式应用就像是真正的天气系统，对它们进行调试如同预测天气一样具有决定性。抖动引入了更多的随机性，因为出人意料，数据可能会逐渐堆积起来。

比如说缓存过期。对于一个热门视频来说，YouTube会尽可能地缓存数据。最热门的视频会

缓存24小时。如果所有缓存同时到期，那么每台机器都会同时计算过期时间，这就造成了惊群。

随机将过期时间设置在18~30个小时之间以实现抖动。这样可以防止数据的堆积。YouTube在所有的地方使用都这种方式。在操作排队时，系统很可能会自我同步，然后摧毁自己。这很值得关注。如果有一台机器的磁盘系统缓慢，然后每台机器都再等待一个请求，这一瞬间在其他机器上的其他所有的请求都会被同步。当你有很多机器，并且同时需要在这些机器上运行很多任务时，这种情况就会发生。每台机器其实都在降低整个系统的熵，因此必须将它加回来。

欺骗：如何伪造数据

这是很棒的技术。最快的函数调用就是不做任何调用。如果有一个单调的增长计数器，比如电影观看次数或是个人资料查看次数，可以在每次更新时处理。或者可以隔一会儿处理，时间间隔随机，只要数字改变是不规律的，人们就会认为它是真实的。这就是伪造数据。

可扩展的组件：创建自己的财富

观察API，做到心中有数：输入是否被很好地定义？将会输出什么？这些都是和数据有关。每个函数中数据的输出及如何流动都要有严格的规范，这将在没有文档的情况下帮助理解相关应用。这样就可以明白函数调用前后发生了什么。

在Python中，数据往往会向RPC移动：代码结构将取决于程序员的能力。因此要做好约定，在其他所有方法都失效时，RPC能助你一臂之力。

没有完美的组件。组件可以持续运行1~6 个月，但具体时间谁也不知道。当情况变糟时，可以把它抽取出来做更改。有时要用Python和C来重写，有时则意味着弃用。只有真正发生，你才能知道是怎么回事。

团队可以有很多人，但没有人能够了解整个系统：因此你需要定义组件。视频转码和视频搜索完全不同，你需要定义良好的子组件。这才是良好的软件设计。这些组件需要互相通信，因此良好的数据规范将会非常有用。YouTube团队做的最糟糕的一个决定就是在Servlet层和模板层之间使用字典作为沟通手段。这是一个非常糟糕的主意。假设添加一个观看页，其上有一个视频，一些评论，还有一些相关视频。这就会导致很多问题，因为字典会有几百个属性。YouTube团队也不是总能做出正确决策的。

用效率换取扩展性

牺牲效率提升可扩展性：效率最高的方式当然是用C来写，然后塞进一个进程中。但这样做就没有扩展性了。

着眼于宏观层面、组件和以及如何划分组件：是使用RPC好还是内联好？把它们分进不同子包，某天可能会发现它们的区别。

重视算法：在Python中实现好算法的可能性很低。比如说这是个二等分的模块，在模块中可以获得一个列表，做一些有意义的事情，然后序列化到磁盘再把它读出来。相对于C来说，这是Python的劣势，但实现起来非常容易。

度量：在Python中，度量就好像是在做审查，因为Python中的很多东西都不直观，例如垃圾收集的成本。YouTube应用的大多数区块将时间花在了序列化上。描述序列化很大程度上取决于放进去的是什么。序列化中断和序列化大对象完全不同。

Python的效率，知道什么不该做

更多是知道什么不该做。你所运行的Python应用的效率与性能消耗与你实现的数据动态程度成正相关。

Dummer代码比grep更简单，也更易于维护：代码越神奇就越难想清楚它是如何工作的。

YouTube很少做面向对象设计：使用大量的名称空间，用类来组织数据，但很少做面向对象设计。

代码树会什么样子的：YouTube团队希望用简单、务实、优雅、正交和可组合等字眼来描述代码树。但这是理想化的状态，现实会有些不同。

（感谢Todd Hoff的翻译授权，原文链接为http://highscalability.com/blog/2012/3/26/7-years-of-youtube-scalability-lessons-in-30-minutes.htm）

MPD：与众不同的研发管理教育模式

当前，软件和信息技术服务业正面临着重大转变，全行业都在加大研发投入，加快企业创新，提升管理能力，企业领导者迫切需要把握趋势、理解需求、重视管理。

正因如此，在2010年，麦思博（msup）创建了软件研发领域团队管理峰会——“MPD”。MPD的名字来源于Make（改变）、Professional（专业）、Discovery（探索）这三个单词的缩写。

在MPD创始人刘付强看来，中国传统的教学、公开课和业界会议，无法实现既能做趋势分享，又能在案例上进行深度演绎。这使他萌生了创建MPD的想法。他希望参会者可以在一个专场学习，完成工作岗位能力胜任模型。

刘付强拿产品总监的角色举了个例子：该职位的学员将在“产品创新/用户体验”分会场两天学习，与模拟方队交流和探讨、思考与共鸣，以完成从思维视野、过程改进、管理支撑、实验室等四个岗位的胜任维度。他介绍，整个MPD项目总共有五个方向：“测试管理/质量平台”、“产品创新/用户体验”、“开发管理/流程再造”、“团队管理/组织发展”和“架构设计/技术战略”。但在各个区域举办时，这五个方向的内容会有不同侧重点，例如在北京地区侧重互联网研发；深圳地区侧重通信和嵌入式研发；二线城市偏重传统研发和成本控制。就区域需求来说，一线城市更希望一些前沿、国际化的案例，二线城市更希望多一些中国本土的研发案例。

MPD的课程设计指标很有特点。刘付强说，在过去的几年，他们曾尝试把每个Skill模块控制在1小时，但学员的反馈是案例深度不够，对工作没有直接帮助。因此，他们按照msup公开课的设置惯例，将MPD项目每个课题控制在180分钟，5个情景角色专场并行2天完成20个Skill模块。这样的举措，可以确保学员在两天里自定义Skill模块，而且可以根据自己的工作环境优化学习进度，进行“穿越”学习；如果企业选派团队参加，就可以平行推进角色设计的课程地图。目前，70%以上的参会人员都是5人以上的团队出席。

截至目前，MPD大会已连续举办了九届。在办会的同时，刘付强也不忘思考，从中总结出了当前中国企业团队发展的困惑。他认为，目前大多数研发团队亟需了解从技术走向管理、走出作坊迈向高绩效研发团队的管理知识，其中围绕流程再造、架构设计、质量管理、产品创新以及组织发展更是当今中高层管理者迫切任务。例如，大部分独立软件开发商是以部门为导向开展工作的，因此，他们必须学会如何采用精益软件架构的技术战略，用敏捷开发和微创新管理品牌。

此外，企业级软件开发对外包管理中面临着需求管理和核心架构设计的问题，这方面的工作必须通过具备相关技能的人才能完成。尤其那些是在中国进行扩建和升级的跨国研发中心，迫切需要培训或招聘具备开发质量意识、熟悉过程改进、深刻理解业务需求的中高端研发人才。

改变互联网的IaaS服务

文 / 王旭

4月10日早上，我和老婆提起，“今天的大新闻是Instagram被Facebook收购了， 10亿美元，iOS平台上最受欢迎的拍照分享服务”，我故意顿了一下说：“这家公司现在只有13个人。”老婆立刻惊问：“他们怎么做到的？”确实难以置信，冯大辉曾在《Instagram架构笔记》里提到，“只有3名工程师，所以自己部署机器到IDC是不靠谱的事情。幸好有亚马逊。” 是的，是亚马逊的IaaS服务—— AWS改变了游戏规则。

本文希望从技术角度，介绍IaaS服务商借助哪些核心技术提供了这样的能力，而创业者如果希望将来能和Instagram一样，服务上千万的用户、存储上亿张照片，应该如何更合理地使用IaaS云服务。

互联网创业的后AWS时代

互联网从业者都知道，在AWS上的成功案例远不止这一家，Dropbox、Zynga的成功离不开AWS的功劳，同样受益的还包括Foursquare、Quora等。正是AWS，或者说由于靠谱的IaaS服务的存在，才为这些创业公司的成长铺就了道路，以至于业界有了“后AWS时代”的提法，后AWS时代包含下面一些特征。

- **随时可用的资源池**。服务上线之前，创业者不需要租用IDC的机架位，也不需要购买服务器，更不需要去机房插网线，当服务需要上线时，在资源池里租用刚好够用的服务即可，没有固定资产投入，运维投入也是从服务上线部署才开始的。

- **快速弹性**。如果仅仅是方便的主机资源，那么也就不需要所谓“云计算”了，VPS们已经做得很好了。但Draw Something只用了几周就有每日1500万活跃用户和每秒3000张图片，Animoto需要在3天之内将主机从5台增长到5000台，这样的规模和扩展速度，不仅依赖于网站的良好架构，也依赖于基础设施服务的高弹性。换个角度考虑，如果你期望自己的服务也可以用这样梦幻的速度增长，却不希望一下就准备这么多设备，具有快速弹性的IaaS服务无疑是首选。

- **用量计费**。有了上面两项特性，相应的成本也需要细粒度地按照用量来计算才划算。对于主机，只需要为运行的时间付费（随时可以新申请或释放掉主机资源）；对于存储，只为使用的空间和流量付费；对于网络，只需要关心实际使用了多少流量。这样更能为小型、刚刚起步的服务节约成本。

- **按需自服务**。所有上面的特征都依赖于便捷的使用，IaaS服务不仅可以通过页面点击完成资源的申请和释放，而且可以通过API，直接编程进行资源操作，配合自助监控服务，IaaS用户可以根据服务的负载情况，自动增加或删除使用的资源，既保证终端用户的体验，又无须无谓付出成本。

新的互联网创业模式正是建立在这些特征之上，它们可以更专注于应用的开发，低成本地提供服务。下面我们来看具体提供这些服务的核心技术，以及如何使用这些技术和服务，确保能够符合预期。

弹性的基础设施

IaaS的基础设施提供了从计算到存储的一整套和传统计算机平台完全一致的环境，从而最大程度地方便了既有应用向云平台的迁移，但云平台也不是“银弹”，并不是一切传统应用迁移到云上就可以无限扩展了。要想引爆云平台带来的弹性，还需要可扩展的网站架构以及合理利用云基础设施的各项实用功能。

面向扩展的服务架构

成为“合格”的云应用，服务架构自身的体质是最重要的。可扩展性，或者说弹性、可伸缩性，追求的是当服务节点规模增加时，服务能力也可以线性或近似线性地提升。这就需要服务请求可以被良好地、互不干扰地分配到多个节点上执行。

简单来说，我们可以把一般的网站架构分成“服务”和“数据”两个大的层次，服务层接纳用户的访问请求，执行各种计算任务，从数据层获取状态信息，并将可能存在的处理结果持久化到数据层；而数据层则要保证数据的正确性、一致性和可用性。

为了线性扩展，服务层就必须做到无状态，这样，任何新上线的节点都可以分担同样的负载请求。同时，要对数据和对数据的访问进行均匀的分片（sharding），在著名的eBay分布式实践（http://www.infoq.com/cn/articles/ebay-scalability-best-practices）中，强调任何数据都要被分片，并避免分布式事务。这样，当面临海量请求时，才可以分而治之，而不是互相影响导致崩溃，重蹈火车订票系统的覆辙。尽管“分治”的思路很简单，但要做到均匀且可动态扩展却并不容易，这一方面需要合理选择划分依据，另一方面需要数据存储层面进行配合，后面的内容还会继续介绍多种云存储和分布式存储技术。

虚拟化技术：弹性的基础

我个人并不赞同过度夸大虚拟化对云计算的意义，显然虚拟服务器上的服务不能被称为“云服务”，但也无法否认，正是日益成熟的虚拟化技术，才让云计算的IaaS服务真正可以成功地大规模投入商业运营。

计算机科学家David Wheeler的有一句名言：“计算机科学中的一切问题，都可以通过再增加一个中间层来解决”，虚拟化计算技术就是这个在多任务操作系统出现之后，又在操作系统和硬件之间插入的中间层，它解决的问题包括以下几点。

- 一台计算机的处理能力已经足够满足多个用户的需求，但在分给多个租户时，他们需要的操作系统环境不尽相同，也需要彼此安全地隔离。
- 动态地调整一台计算机（或者说操作系统）的物理资源不够容易，至少不容易自动化。
- 安装或迁移一台服务器需要大量的时间，而且不灵活。

虚拟化技术主要是一种隔离与监视技术，让不同的虚拟机分享同一套物理机（主要指CPU/内存子系统）而彼此隔离、互不干扰，同时，为虚拟机代理访问网络、磁盘I/O等共享的外部资源。由于虚拟化技术对CPU/内存子系统工作的干扰很少，因此，计算能力方面很少有开销，相比之下，I/O性能受到的影响会略大一些，但也可以接受。

目前的IaaS服务中，主要的虚拟化技术包括开源的Xen、KVM，以及VMware的ESX和微软的Hyper-V。它们的工作原理都比较接近，以Xen为例，它通过一个Hypervisor来监控管理多个不同的虚拟机（称为Domain），并将I/O等访问特权资源的操作在专门的特权域（Dom0）中执行，其他运行客户操作系统的虚拟机称为用户域（DomU）。某些DomU操作系统是被修改过的操作系统，它们知道自己运行在虚拟化环境中，并不直接和硬件打交道，于是，它可以将IO请求通过“XenBus”直接发送到Dom0，由Dom0代理完成，这种方法效率很高，称为半虚拟化（PV，ParaVirtualization）。对于不支持以PV方式运行的虚拟机，我们需要首先实现一个虚拟的设备（在Linux下，Xen和KVM的虚拟设备都是基于Qemu的），提供给虚拟机，它再将这些IO请求代理发出，这种方式的IO相比之下更加繁琐、效率更低，称为完全虚拟化（FV，Full Virtualization），优点是可以兼容任何操作系统、无须修改。

实现快速扩展的公有云运维技术

即使虚拟机可以快速分配、部署了，用户服务程序的部署也不见得是一个简单的过程，尤其要在服务提供的过程，自动化地扩充服务器，更需要费一番力气。实际上，除了使用配置服务等“传统手段”之外，IaaS服务商们还提供自己的一些运维用工具，帮助用户更好地实现弹性服务。

这些技术之一就是快照/镜像服务，很多云服务商都已经提供了这项功能，它可以帮助用户保存虚拟机的一个快照性质的副本，一旦用户的虚拟机

发生故障，可以用快照恢复；同时，因为用户的应用服务器一般是无状态的，这样，通过定制镜像可以直接快速复制出大量包含用户应用的、相同配置的虚拟机。而且，对于实现良好的IaaS服务，镜像服务系统可以加速热点镜像的传播，进一步加速了用户服务的扩展。

通过API实现自动扩展

IaaS的目标之一就是帮助用户尽可能降低运维的复杂度，即使用户需要随着服务规模的增长迅速扩展服务能力，仍然能够举重若轻。这里，不可忽视的一个方面就是云服务的API，对于一个“可编程”的IaaS服务，用户可以在业务需要时，通过RESTful API请求来直接创建新的虚拟机，而无需登录管理页面。

更进一步，与监控程序配合，用户的扩展脚本可以根据性能监控的数据，实现业务流量在一定范围内发生变化时，无人值守地扩展或缩小服务规模，既不浪费资源，又不影响服务质量。通过云监控、API和扩展脚本，以及上面提到的镜像服务的配合，用户可以实现高效可靠的自动化扩展能力。

善用存储服务

前面提到数据层对于网站可扩展性的重要性，这里就展开介绍云中的各种存储服务。存储本身说来简单，但实际需求却很多样，有提供客户端访问的，也有服务自己的元数据；有需要持久化的，也有用于加速的缓存；有要求严格一致的交易数据，也有不太重要的状态更新；有结构化的，也有非结构化的。而不同的存储系统、存储服务也是为不同用途所准备的，要真的做到高效、低成本、可扩展，就需要为应用选择合理的存储服务。

对象存储服务

对象存储服务是指可以通过HTTP RESTful接口的API，以键值访问的数据对象。典型的对象存储服务是亚马逊的S3服务以及国内相应的云服务商的云存储服务。通常对象就是一个文件，如图片、文档、音乐、压缩包等，而用于标识对象的键值可以看作是文件名。对象被组织在称为桶（bucket）的容器中，每个桶可以指定特定的访问权限等属性。与网盘或分布式文件系统不同，对象存储服务不提供全局的命名空间和复杂的目录树，只有桶–对象二级结构，这既简化了系统的实现，也提高了访问效率。当然，要把直接使用既有服务迁移到对象存储服务上来，需要进行一定的调整。

对于使用IaaS云搭建的云服务，推荐使用云存储服务存放需要浏览器或是客户端直接下载的静态内容，因为这样一方面可以直接使用云服务带来的高可靠、高可用性，利用云服务商提供的网络加速功能；另一方面，相对于虚拟机存储空间的局限性和其他存储服务的成本来说更低廉，也不会占用虚拟机服务的带宽，是一种性价比更高的存储服务。

云硬盘服务

云硬盘服务或弹性块存储服务（EBS，Elastic Block Store）并不是网盘，而是一种提供给弹性计算中的虚拟机使用的，具有独立于虚拟机的永久存储能力和高可用性的存储服务。对于虚拟机来说，云硬盘使用起来就是一块硬盘。用户可以申请一块任意大小（一般是1-1000之间的整数GB）的云硬盘，将它挂载到自己的任意虚拟机上使用，也可以将它在一台虚拟机上卸载，并挂载到另一台虚拟机上，甚至于可以删除所有虚拟机，只保留云硬盘。

相对于虚拟机自己的存储空间，云硬盘的大小更灵活、可以由用户指定；云硬盘具有更高的可靠性，即使虚拟机发生故障，云硬盘由于有后台的类似RAID的多副本保障，仍然可以保障数据的可靠性。云硬盘的生命周期与虚拟机彼此独立，弹性计算服务鼓励用户在不使用时，释放虚拟机，让出资源，同时也节约自己的开销，这样的情况下，云硬盘可以承担数据持久化的功能，不必因为释放虚拟机就丢弃数据。

尽管作为一块硬盘，云硬盘显然可以存放图片、附件等内容，但并不意味着云硬盘是比对象存储更好的存储系统。两者是面向不同用途的，如上所说，使用对象存储意味着更低的成本和往往更

好的客户端访问性能，可以利用服务商的网络加速能力，并且不占用虚拟机的网络流量；作为对比，云硬盘可以从虚拟机更快地访问（读取操作平均响应时间一般是小于10毫秒的），各种程序可以像访问本地磁盘一样使用云硬盘中的数据，当然，云硬盘会比云存储贵一些。因此，相对于面向客户端请求的对象存储服务，云硬盘更倾向于存放数据库等供虚拟机直接使用的持久化数据。

云中的关系型与非关系型数据库

毫无疑问，Web 2.0时代的基石之一是一种开源的关系型数据库——MySQL，但随着移动互联网和社交网络的发展导致的数据爆炸性增长，关系型数据库的范式模型和事务性限制了它们在保持原有功能性的情况下，水平扩展到更多节点上。于是，在大数据时代，很多非关系型数据库（NoSQL）开始受到越来越多的推崇。

根据Brewer的CAP原理，对于一个分布式系统，一致性（Consistency）、可用性（Availability）和分区容忍性（Partition Tolerance）3个特性，只能强化其中两个。作为面向大数据的分布式存储系统，分区容忍性是必须的选择，因此，就不得不在一致性和可用性之间抉择，选择一致性的系统不得不在有些情况下无法提供服务，而那些强化可用性的服务则允许系统在某些时刻不完全一致，也就是所谓的“最终一致性”。用户在选择一个存储系统时需要选择，自己的数据模型怎样、是否需要强一致性。当然，NoSQL和MySQL之间并非取代关系，两者各有优势。可以负责任地说，大部分应用还没到MySQL无法承受的数据规模，只是有的应用需要更良好的架构设计而已。

而对于NoSQL，也有很多可以选择，比较流行的包括面向JSON文档的、灵活的MongoDB，主要用于内存缓存的、高性能的、通过键值方式访问的Redis，基于Hadoop、提供列族模型的HBase，以及同样提供列族模型的、写入性能很高的、无中心的Cassandra等。

很多IaaS云服务商也提供了SQL与NoSQL的增值服务，用户可以选择安装、调优好的虚拟机，作为云数据库服务。

CDN服务

CDN（内容推送网）服务很难确切定位为存储还是网络服务，它用于加速用户的网络访问，其加速手段是通过部署在接近终端用户位置的缓存服务器，向他们提供高速内容服务。

传统的CDN服务一般是面向访问量很大的客户的加速服务，不仅贵，而且购买、调整不方便。而IaaS云服务商面向它们的客户，提供了自助化的CDN加速服务，使得中小网站也可以享受。

展望：从IaaS到PaaS

虽然相对于IaaS，PaaS还没有大量改变我们的生活，但并不妨碍很多业内人士将其视为公有云的未来。与IaaS的简单直接不同，PaaS服务并不直接提供资源，而是提供运行环境和功能组件（库）以及附加的服务，如数据库、队列及其他特殊功能，更方便进行开发、部署，而将伸缩、扩展等复杂性进一步屏蔽。

在PaaS服务逐渐成熟的同时，一些构建在IaaS层之上的PaaS也开始出现，成为IaaS的增值服务，使得两者的界限逐步开始模糊。此外，一些可以构建在多种IaaS上的“开放PaaS”平台的出现，更进一步方便了IaaS服务商提供附加的PaaS服务，给用户更多的选择。这一演进将进一步降低运维开销，让创业者把精力专注于创新和提高用户体验，实现彼此共赢。

王旭

盛大云计算高级研究员，主要专注于分布式存储系统的开发，加入盛大云计算前在中国移动研究院进行Hadoop相关开发工作。译有《Cassandra权威指南》，个人网站为 http://wangxu.me/blog/。

开源云先锋OpenStack

文 / 杜玉杰

在我刚开始接触OpenStack这个开源项目时，就被它开放的口号吸引了，该项目确实能对构建开放云基础架构产生深远影响。在此之前，云基础架构的主要两大阵营是Amazon API及其兼容架构和VMware的vCloud。作为第三种选择，OpenStack为服务提供商和电信运营商提供了一个开放的平台，至少在这一领域我们需要打破双寡头的垄断，引入具有竞争力的参与者。

开放为何如此重要？

大多数人认为云基础架构互操作性的问题在于磁盘镜像格式（例如VMDK、VHD和qcow）或者是 hypervisor。而实际上，很多磁盘镜像格式是块存储，在VMDK和VHD之间转换并非技术难题，特别是大多数hypervisor都依赖于硬件虚拟化（HVM）的情况下效率不是问题。那究竟是什么阻碍了不同云之间的数据迁移呢？简言之，是架构不同。如果每个云都有自己的存储和网络架构，就会造成人为的困难。

一直以来，对于亚马逊网络服务及其他云服务抱怨最多的就是“导致用户被绑架，无法轻易转移数据”。在云计算社区，有一个流行的概念，即数据是有重量的，一旦你将数据存在某个云计算提供商那里，它就变得繁重而难以迁移。OpenStack构建了一个强大的社区并足以推动实现一个标准化的可互操作的参考架构。

总之，OpenStack为希望像AWS和Google一样提供开放云基础设施的企业提供了一个可行路径。

世界变了吗？

2010年，OpenStack项目刚刚启动时就已经有人在说：“OpenStack, with a strong community behind it, should be an important tool for service providers and large telcos to compete at scale with the Amazon and Googles of this world.”

两年的时间过去了，现在这个“strong communtiy”在全球已经有20多个用户组，近3000名成员，以及165多家企业，其中既有初创公司也有跨国企业，并且有55家以上的企业大约200名开发者贡献了代码，包括来自中国的新浪[1]。

但它是否已经成为“an important tool for service providers and large telcos to compete at scale”了呢？目前来看毫无疑问的是，像KT、Internap和HP等都已经部署了OpenStack平台，而所发生的这一切只有不到两年的时间。

现在，OpenStack社区正在朝着成为自Linux以来下一个最有影响力的开源项目而迈进。由19家公司组成的OpenStack基金会已经成立，标志着OpenStack社区将会有长期的财政支持，并开始独立运营。

还是“那个”OpenStack吗？

OpenStack社区于2012年4月5日顺利发布Essex

版，该版本最大的特色就是“production-ready”，主要包括以下五个组件[2]：

- Nova[3]：对应于Amazon EC2，控制IaaS；
- Swift[4]：对应于Amazon S3存储，对象存储；
- Glance[5]：处理虚拟磁盘镜像，支持qcow2（KVM）、AMI、VMDK（VMWare）和VHD（Hyper-V）等格式；
- KeyStone[6]：提供通用的身份管理服务；
- Horizon[7]：提供基于Web的管理服务。

Essex的主要新特性有以下几个方面。

- Swift的改进

支持对象过期时间，例如可以设置文件在某个时间点过期，可用于文档管理系统根据策略保留文档。还增加了中间件模块允许开发者创建控制面板直接上传文件到Swift。

- Nova的改进

增加了基于角色的访问控制（RBAC），Nova目前可以支持对动作和扩展的可配置的权限控制，改善了orchestration特性，同时也提高了安全性。

- 集成Keystone

Keystone是OpenStack的通用身份管理系统，有两大功能：基于token的认证（authentication，即authN）和基于用户-服务的授权（authorization，即authZ），可以连接外部认证系统，如LDAP。

- Volume的改进

支持从Volume启动，虚拟机根分区不需要本地存储，为以后的虚拟机迁移提供了方便。

支持Volume的快照备份API。

- Glance的改进

同样，Glance也支持RBAC，保护镜像文件，以防止意外删除虚拟机镜像文件。并且采用可配置的进程数以避免之前受限于单一CPU而带来的性能问题。

- 其他改进

API默认支持https。

对OpenStack中组件之间的消息通信加密。

与Amazon相比，OpenStack要开放很多

数据库权限分离，每个API Server使用自己的数据库账号，并且只能管理自己API所关心的数据表。

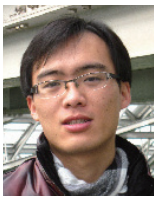
更加友好的Dashboard。

小结

从Essex版本的这些组件和特性看出，随着OpenStack的不断完善和发展，必将成为开源云计算的一大先锋。

参考资料

1. http://www.openstack.org/projects/essex/press-release/
2. http://wiki.openstack.org/ReleaseNotes/Essex
3. OpenStack Compute ("Nova") 2012.1，网址为：https://launchpad.net/nova/essex/2012.1
4. OpenStack Object Storage ("Swift") 1.4.8，网址为：https://launchpad.net/swift/essex/1.4.8
5. OpenStack Image Service ("Glance") 2012.1，网址为：https://launchpad.net/glance/essex/2012.1
6. OpenStack Identity ("Keystone") 2012.1，网址为：https://launchpad.net/keystone/essex/2012.1
7. OpenStack Dashboard ("Horizon") 2012.1，网址为：https://launchpad.net/horizon/essex/2012.1



杜玉杰

OpenStack中文社区管理员、社区经理，目前就职于中标软件有限公司。

PaaS——云计算的下个制高点？

文 / 方国伟

PaaS平台的理解

云计算服务模式的比较

大家知道，云计算主要是通过构建共享资源池的方式来提高资源利用率的。在云计算当中，根据这个资源池中资源的类别，我们把云计算的服务模型分为软件即服务（SaaS）、平台即服务（PaaS）和基础设施即服务（IaaS）三种。不同服务模型的服务供应商所提供的服务存在较大的差异。从应用运行堆栈的角度看，三种服务模型与传统IT方式比较如图1所示。

图1 云计算服务模型比较

云计算的三种服务模型所针对的用户类型实际上是不一样的。由于SaaS提供的是应用程服务，因此它针对的是最终用户，也就是一般的应用程序使用人员。用户在使用SaaS时一般不需要对技术本身有特殊要求，只要了解软件服务本身的一些操作规则即可。用户一般也不需要对服务商提供的SaaS服务进行二次开发，只需要简单设置就可以使用。PaaS提供的是平台服务，因此它针对的用户是开发人员。PaaS需要开发人员针对其平台的编程接口进行应用程序设计和开发，然后部署于其上。如果这个部署的应用是对外向多个租户提供软件服务，那就是SaaS服务。IaaS提供的是最底层的IT基础设施服务，因此它直接针对的用户是IT管理人员。IaaS提供的服务首先需要IT管理人员来进行配置和管理，然后才能在其上进行应用程序的部署等工作。

SaaS的好处在于对用户的要求比较低，基本是提供现成的应用服务供用户直接使用，但问题在于，用户的需求不是都有现成应用服务来解决。PaaS的好处在于用户可以直接在上面开发自己需要的应用程序，而不用关心底层系统平台的维护，但它的问题在于不利于已有应用的迁移。IaaS对已有应用的迁移相对比较平滑，用户使用平台的灵活度相对要大一些，但用户要做的事情也相对更多。因此，如果从自动化和灵活两个维度来分析SaaS、PaaS、IaaS及虚拟化这四种服务方式，情况如图2所示。

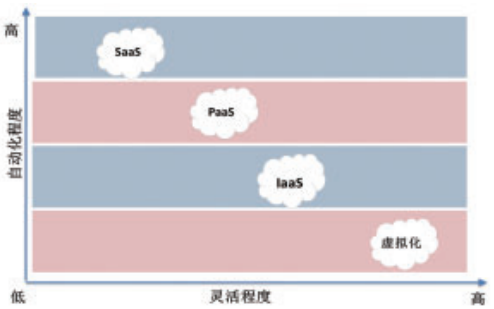


图2 服务模式的自动化和灵活度对比

可以看出，在云计算的三种服务模型中，SaaS自动化程度最高，但灵活度最差。因为服务供应商提供什么服务，用户就使用什么服务，所以不用也不能做太多别的事情。IaaS灵活度有余但自动化程度较低，要求用户自己做许多事情。而PaaS是一种折中选择，在自动化和灵活度两个方面做了平衡。

PaaS平台的理解

从服务模型的功能定位来看，PaaS给应用提供了一个托管运行的环境。当然这个应用托管环境是需要按照云计算的要求和技术来实现的，因此平台需要提供自服务、多主租用、资源按需动态分配、弹性扩展、按照使用量计费等功能。开发人员只需要针对平台开发应用程序即可，而不用关心底层平台的具体情况，比如平台安全、系统升级、补丁等。从应用运行层次的角度看，PaaS可以被认为是传统的操作系统和中间件功能的一种集中体现。通过使用PaaS平台，开发人员可以把精力放在他们的应用逻辑上而不是在部署和管理云服务的基础架构上，并且可以节省许多开发、部署应用的时间和费用。

从功能需求的角度，一个完整的PaaS平台需要包含下面一些内容。

■ 提供计算服务

PaaS平台需要为应用提供分布式计算功能，应用的计算能力可以根据应用实际计算需求和成本进行动态调整。计算能力能够进行横向水平扩展。不同的PaaS平台会根据自身特点选择采用在虚拟化基础上实现或直接在物理节点上构建应用的计算环境。

■ 提供数据存储服务

计算是围绕着数据进行的，因此数据存储服务是PaaS平台必不可少的一个组成部分。PaaS平台可以提供一种或多种不同的数据存储方式，比如传统的结构化数据存储，或针对非结构化数据的NoSQL类型数据存储服务等。

■ 中间层及其他辅助服务功能

与传统企业计算的中间件类似，为了加快应用开发的速度并降低应用开发难度，PaaS平台往往会提供一些常用的中间层服务，比如身份认证、服务总线、工作流等。另外，PaaS平台对于大访问量的应用还会提供分布式缓存、负载均衡和CDN等服务。部分PaaS平台还会进一步提供一些接近应用层面的服务，比如邮件集成服务、搜索服务等，从而使得平台用户可以在这些服务的基础上构建组合式应用。

■ 开发工具和测试环境

一般PaaS平台上托管的应用都需要针对平台本身进行开发和定制，因此完整的PaaS平台往往会提供配套的开发工具、软件开发工具包（SDK）和本地测试环境等，从而方便平台用户开发应用。大部分PaaS平台供应商都是通过为常用的开发工具如Visual Studio、Eclipse等提供插件的方式来实现。

PaaS平台分类和发展

当一个应用发布在PaaS平台上进行托管运行的时候，它会使用多种不同的服务。比如根据应用开发时所使用的编程语言不同，应用需要依赖不同的运行时或运行环境（容器）。应用还需要存储数据，因此需要使用数据存储服务。如果不同的应用服务需要集成和交互，那可能还需要用到总线服务或流程引擎服务等。目前包含所有这些服务的PaaS平台还比较少，大部分都提供其中一种或几种。由于这些服务大多是松耦合结构，因此PaaS平台的构建往往也是分期构建并逐步完善的。

根据平台提供服务的不同，PaaS平台可以分成不同类型。比如有的PaaS平台侧重于提供关系型数据存储，也称为Database as a Service（数据库即服务），有的平台侧重于提供给予服务总线的集成，也有的专门提供身份认证。Gartner在2011年发布过一个关于PaaS平台发展路线的报告，归纳了PaaS平台类型及大致发展路线，如图3所示。

我们可以看到图3中的PaaS平台划分基本是按照IT层面的需求来进行的，我把它归结为通用类型或水平类型的平台服务。现实的平台层服务需求中还有一类是专业类型或垂直类型的平台服务。比如对于一些新兴的社交类互联网公司，它对提

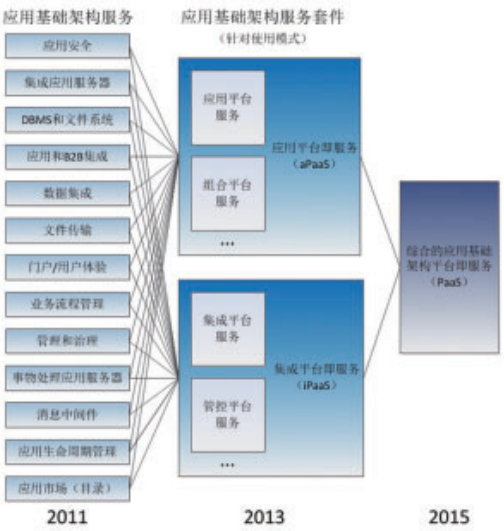


图3 Gartner发布的PaaS发展路线

供通用的应用托管运行环境兴趣不高，更关心的是如何在平台层面提供与自己社交相关联的服务，从而让第三方开发人员用这些服务开发出更多应用。这样可以大大提高这些社交服务的黏性和使用范围，并逐步把自己的社交服务向社交平台方向发展。另外一个例子就是像电信运营商这样的传统专业服务公司。虽然一些电信运营商也提供托管服务，通用类型平台服务对它们的传统托管服务也是一种提升，但这些平台服务不能充分发挥它们自身的业务优势，因此不容易在市场上打造自己的竞争优势。如果电信运营商能够把它们的电信业务通过平台层服务开放出来，那么就相当于通过云计算模式拓展了它们传统的电信业务。这种能力是一般通用类平台服务提供商所不具备的，从而可以在市场上形成自己独特的竞争力。

云计算服务从IaaS发展到PaaS，从本质上看也是服务差异化的一种体现。因为IaaS层面服务的竞争到最后慢慢会进入到同质化阶段并陷入成本的竞争。同样，在PaaS层面提供与自己行业相关的平台服务，实际上也是服务差异化的一种体现。不同行业中具有创新能力和实力的公司会逐步按照这个方向发展。

业界PaaS平台的发展情况

在云计算刚开始发展时，市场上的云计算服务大多是SaaS或IaaS类型的服务。但随着越来越多的用户开始转向云计算，就出现了新的需求点。当用户越来越多时，用户需求的多样性就表现出来，单一的服务越来越难以满足用户的需求。另外，最初的云计算用户在思想开放性和技术水平上都是相对领先的，但随着越来越多的用户转向云计算服务，这对云计算平台的易用性提出了更高的要求。因此，SaaS服务的单一性和IaaS服务的复杂性就越来越显示出它们的局限。下面我们来看几个主要云计算参与方在PaaS平台方面的进展情况。

微软——Windows Azure

微软一直以来都以提供平台和办公应用产品见长，加上还有许多像MSN、Hotmail、Bing等互联网业务，所以微软在全面转向云计算服务之后提供一个PaaS平台是非常自然的选择。微软的目标就是要构建下一代互联网服务平台，为微软公司自己、客户和合作伙伴提供互联网规模的应用服务平台——Windows Azure云平台。对微软而言，它主要的云计算平台就是Windows Azure，而传统的办公应用就逐步演变成SaaS类型的Office 365。

微软推出Windows Azure的目标是构建一个云计算操作系统。传统操作系统只需要管理一台或几台计算机上的资源，但Windows Azure作为云计算操作系统需要管理一个或多个数据中心中的所有资源。Windows Azure为开发者提供了托管的、可扩展的、按需应用的计算和存储资源，还为开发者提供了云平台管理和动态分配资源的控制手段。开发人员在构建Windows Azure应用时，不仅可以使用不同的开发语言如.NET、Java和PHP等，还可以使用不同工具，如大部分开发者熟悉的Microsoft Visual Studio、Eclipse等。这样开发人员的许多经验和技能都可以相对平滑地从面向传统平台编程转到基于云计算平台的编程。Windows Azure在很大程度上代表了Windows Server和SQL Server业务的未来。

Google——App Engine

我们知道Google是从搜索业务起家，然后逐渐扩

展到提供电子邮件、日历、Docs等多种Web应用。Google后来把这些Web应用打包成不同版本的Google Apps，并对外以SaaS业务方式进行销售和推广。但显然，这些Web应用只能涵盖最为普遍的一些办公需求，因此Google在挖掘新的Web应用的同时推出了自己的PaaS平台——Google App Engine（GAE），从而为用户提供一个平台层服务来开发和部署自己的Web应用。

与Windows Azure相比，GAE平台的抽象层次要高一些，因此它对用户开发应用的限制也相对要多。比如在平台语言支持方面，GAE开始时只提供Python语言的支持，后来添加了对Java和一些可以运行在JVM之上的动态语言支持，但由于GAE托管应用运行在一个“沙箱”里，因此只能支持这些语言功能的一个子集。另外一个差别是，GAE中提供了许多与Google SaaS层Web应用集成的服务，这样可以增加平台层应用的黏性并扩展其SaaS服务的使用。从平台服务层次上看，Google的GAE更靠近SaaS层，而微软的Windows Azure更靠近IaaS层。因此，GAE支持更多的是基于Web的应用，而Windows Azure支持的应用类型更广一些。

Salesforce——force.com

Salesforce是最早的SaaS服务公司之一，它从提供在线的CRM系统开始，逐渐扩展到其他管理软件业务。随着Salesforce用户数的增长，用户对它的CRM等系统的定制要求也越来越高，纯粹的SaaS服务对部分用户已经不够了。因此，Salesforce推出了自己的PaaS平台——force.com。从这个平台产生的背景可以看出，force.com主要是围绕着Salesforce的SaaS业务展开的。force.com专门提供了一种名为Apex的编程语言，允许用户开发托管在平台上的应用。这些应用一般提供可以集成到Salesforce商业应用的功能，从而满足客户对SaaS服务的定制化需求。另外，Salesforce利用自己在运营SaaS的经验把Web应用托管、在线数据库服务（database.com）等都逐步作为平台服务开放出来。

Salesforce正在加强PaaS平台的建设，它不仅基于force.com提供服务还与VMware等厂商合作推出VMforce，提供Java应用托管环境。除此之外，Salesforce还进行了许多并购以加强服务范围，其中包括一家名为Heroku的PaaS平台公司。Heroku最开始是提供基于Ruby的Web应用托管，然后逐渐增加了对Java、Node.js、Python等语言的支持。所以，Salesforce已意识到SaaS服务的局限性，正在利用自己原有的基础设施、运营经验和客户群体向PaaS进行业务扩展。

亚马逊——Elastic Beanstalk

在IaaS层面的服务中，目前最著名的是亚马逊提供的AWS。亚马逊通过不同的方式把自己庞大数据中心的基础设施对外提供出租服务，比如通过弹性计算服务（EC2）提供虚拟机租用，通过简单储存服务（S3）提供存储租用等。传统上AWS的侧重在IaaS层面，亚马逊一直在逐步扩展它的业务范围，其中非常明显的一个特点是从下往上的发展方式。亚马逊最开始向平台方向扩展的方式相对简单，就是在其提供的虚拟机中事先安装配置好各种平台软件，包括常见的中间件软件和关系型数据库软件。

通过预装平台软件这种相对简单的方式能够满足部分用户对平台服务的要求，但这种方式毕竟缺乏整体考虑，平台的灵活性和管控能力都相对偏弱，而且通过这种方式集成第三方的软件平台有许多技术和商务上的限制。因此，亚马逊在2011年推出了AWS Elastic Beanstalk，这是个真正意义上的PaaS平台。这个服务目前还处于Beta版阶段，可以支持基于Java和PHP的Web应用。前者运行在AWS管理的Tomcat服务器之上，后者运行在其管理的Apache HTTP服务器。由于AWS传统强项在IaaS层面服务，因此亚马逊的这个PaaS平台服务从层次上看也是更靠近IaaS。Elastic Beanstalk简化了用户对应用的部署和管理，而且可以非常容易和AWS的其他服务相集成，可以看成是IaaS服务的延伸和扩展。

VMware——Cloud Foundry

一提起VMware大家会立刻想到它的虚拟化技术和产品，但VMware最近几年一直在不遗余力地加强其在云计算方向的实力。通过在虚拟化

的基础上增强其在资源管理、流程自动化管理、自服务门户等方面的能力，从而把自己逐渐转向一个云计算厂商。但与前面几个云计算厂商不同的是，VMware自己目前还没有大规模的商业化云计算服务，它的重点是为客户和合作伙伴提供产品和解决方案。VMware的主要产品在虚拟化领域，因此它的云计算方案主要也是在IaaS层面。

为了拓展云计算解决方案的范围，VMware在2009年8月收购了SpringSource——一家专注于Spring应用框架的厂商。这是从IaaS层面解决方案向平台层解决方案发展的一个标志性事件。一方面是因为VMware进入PaaS市场相对比较晚，另一方面也是因为它在构建平台时采用了大量的开源软件，因此VMware的PaaS策略是推出一个基于Apache License 2.0的开源PaaS平台——Cloud Foundry。除了可以作为软件供用户下载外，VMware自己也运营了一个基于Cloud Foundry的平台服务，目前还处于Beta版测试阶段。Cloud Foundry支持多个以JVM为基础运行环境的应用编程框架，包括Spring for Java、Ruby on Rails、Node.js等。为了向平台层服务扩展，VMware还推出了称为vFabric的集成式应用平台。vFabric在VMware底层虚拟化基础平台之上提供了一个分布式应用平台，提供包括数据存储、消息队列、Web服务器、Java运行环境等功能模块，主要用来支持基于Spring框架的Java企业应用。VMware的平台策略是充分利用在虚拟化基础架构的优势，然后通过以Java应用框架为主的方式向上层服务扩展。

综上所述，除了微软一开始把PaaS平台作为其云计算战略重点之外，前面描述的其他几家云计算厂商都在逐步加强PaaS平台服务。它们的服务类型和客户群体发展趋势可以参考图4。

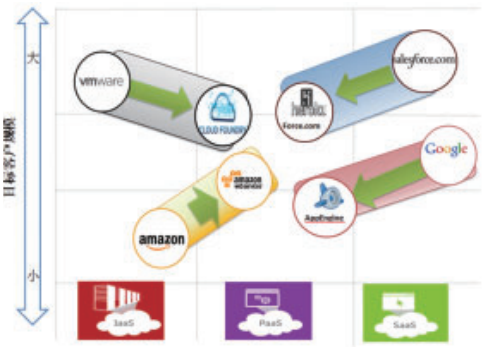


图4 云计算服务发展趋势

结语

对服务供应商而言，SaaS服务的特点带来了两个挑战。一是服务单一性决定了其服务种类范围会有很大的限制，因为每个SaaS类服务都依赖于具体的应用，大多SaaS服务供应商只会提供一个或一类SaaS服务。二是SaaS服务提供的都是标准化服务，这对有差异性要求的用户而言显得不够灵活。对于IaaS服务供应商而言，随着IaaS服务市场的逐步成熟，IaaS服务本身会逐渐陷入同质化阶段的成本竞争，因此如何差异化自己的服务变得非常迫切。一方面服务供应商可以在安全性、可用性、价格等方面进行差异化；另一方面在IaaS服务的基础上提供PaaS层面服务也不失为一个好的发展方向。

PaaS平台服务是一条介于SaaS和IaaS的中间道路，具有IaaS服务的灵活性但在自动化程度上面和易用性方面又超越了IaaS服务。因此，我们会看到越来越多的云计算服务供应商和云计算解决方案厂商在充分发挥原先云计算服务的基础上开始转向PaaS平台。一些原先专注于SaaS的服务供应商开始由上往下发展（可以称之为Top-Down方式），而一些原先专注于IaaS的服务供应商或解决方案厂商开始从下往上发展（可以称之为Bottom-Up方式）。我们已经可以看到PaaS平台正在成为下一个云计算市场争夺的焦点，你准备好了吗?

方国伟

微软全球服务部门数据中心和私有云“卓越中心”资深架构师，负责微软云计算服务和解决方案在中国及亚太地区的推广和支持工作。

利用SAE数据存储技术开发应用

文 / 丛磊

Sina App Engine的发展

新浪研发中心于2009年8月开始对Sina App Engine（以下简称SAE）进行内部开发，它是新浪云计算战略的核心组成部分。2011年5月，SAE正式对外开放注册，成为国内第一个正式开放的云计算平台。

截至2012年3月，SAE上已经有16万开发者，20万应用，覆盖传统互联网和移动互联网领域，成为国内最大的云计算开发运行平台。

众多运行在SAE的网站和应用中，有云存储产品“微盘”，游戏平台“微游戏”，网络游戏“微三国”，还有某银行信用卡部分业务，也有北京外国语大学等教育机构，更有诸如像“你画我猜”这样的热门移动应用。

SAE实际是一个开发、运维、管理平台，目前支持PHP、Java、Python这3种Web开发语言，其中Java和Python处于公测阶段，Web开发者可以在Linux/Mac/Windows上通过SVN、SDK或者Web版在线代码编辑器进行开发、部署、调试，团队开发时还可以进行成员协作，不同的角色将对代码、项目拥有不同的权限。

SAE提供了一系列分布式计算、存储服务供开发者使用，包括分布式文件存储、分布式数据库集群、分布式缓存、分布式定时服务等，这些服务将大大降低开发成本。同时又由于SAE整体架构的高可靠性和新浪的品牌保证，降低了开发者的运营风险。运维人员可以通过SAE的日志统计中心观察网站运行情况，了解资源使用情况和性能瓶颈。其中，作为典型的云计算，SAE采用“所付即所用，所付仅所用”的计费理念，精确地计算每个应用的资源消耗（包括CPU、内存、磁盘等）。

数据存储服务

从技术的角度，对于运行在SAE上的千千万万应用来说，影响最终用户体验的最关键因素就是数据存储。而数据存储服务最重要的技术点就是扩展性、读写速度和可靠性。本文就从这3个角度介绍SAE提供的主要数据存储服务。

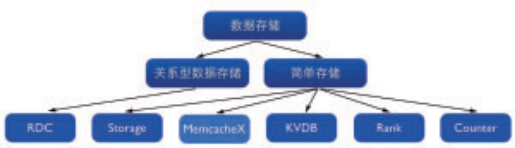


图1　SAE数据存储服务分类

图1列出了SAE提供的主要数据存储服务，在传统关系型部分，SAE提供MySQL服务，而MySQL服务是由SAE自己研发的分布式数据库集群RDC提供的。Storage是一套分布式文件存储服务，满足用户在分布式环境下读写一致。MemcacheX是SAE基于Memcached开发的分布式缓存服务，完全兼容Memcached协议，使得用户在使用时没有任何障碍。KVDB是SAE研发的一套key-value分布式数据存储服务，它的特点是无限扩展并支持无缝动态平衡。Rank也叫实时排行榜服务，开发者可以通过它来满足一些特殊场景的排名需求，比如游戏应用的玩家top排名。和Rank类似，Counter计数器服务也是特殊场景需求的数据存储服务，开发者利用它来完成诸如评论计数等需求。

在这些数据存储服务中，MemcacheX是非持久化存储服务，而其他均为持久化存储服务。

MySQL/RDC

最近对关系型数据库的争议比较多，主要是针对它的扩展性。因为技术原因，目前的关系型数据库基本不具备很好的扩展性，即使有些数据库特性里有高可扩展性，也是通过增加用户学习成本来完成的，比如加入partition-key的概念，让开发者指定分区规则。SAE在这部分采取了这样一个概念“应用开发者负责和自己业务相关的高扩展性”。比如数据库分库分表，这样的区分往往和实际业务逻辑相关，所以SAE把这部分交给开发者自己负责。

SAE默认每个App一个数据库，用户如果需要一个App对应多个库，有两个办法。

1. 建立多个应用，然后进行MySQL跨应用授权，将这些应用的库授权给某一个应用A使用，这样相当于应用A就拥有了多个库。

图2 MySQL跨应用授权设置

MySQL跨应用授权还有一个作用，提高用户MySQL的安全性，即使用户的accesskey和secretkey（实际对应为MySQL用户名和密码）被别人窃取，只要你的数据库没有授权别人的应用访问，别人仍然无法访问你的数据库。

2. 申请SAE企业版，在SAE企业版，开发者将可以建立多个库。

SAE支持多表，目前一个库内的表数量最大为512个，开发者在开发时，可以按照自己的业务逻辑（比如按UID哈希、按时间轴递增）进行分表。

非业务相关的扩展性RDC天然支持，这里主要是指读写分离。SAE PHP提供MySQL封装类SaeMysql自动支持读写分离，PHP环境习惯操作原生MySQL接口的用户和Java、Python环境的用户，也可以通过获取环境常量执行MySQL操作：

```
用户名   :  SAE_MYSQL_USER
密   码 :  SAE_MYSQL_PASS
主库域名 :  SAE_MYSQL_HOST_M
从库域名 :  SAE_MYSQL_HOST_S
端   口 :  SAE_MYSQL_PORT
数据库名 :  SAE_MYSQL_DB
```

对于PHP开发者：

```
$link=mysql_connect(SAE_MYSQL_HOST_M.':'.
SAE_MYSQL_PORT,SAE_MYSQL_USER,SAE_MYSQL_
PASS);
if($link)
 {
  mysql_select_db(SAE_MYSQL_DB,$link);
  //your code goes here
 }
```

对于Java开发者：

```
String url="jdbc:mysql://w.rdc.sae.sina.
com.cn:3307/app_myappname";
String sql = "INSERT INTO app_table VALUES
(1,'110101')";
String username=SaeUserInfo.
getAccessKey();
String password=SaeUserInfo.
getSecretKey();
String driver="com.mysql.jdbc.Driver";
Class.forName(driver).newInstance();
Connection con=DriverManager.getConnectio
n(url,username,password);
```

从上面的示例可以看出，因为RDC完全兼容MySQL5协议，所以用户通过原生MySQL驱动即可以操作MySQL。实际上所有SAE开发者看到的MySQL主库地址和MySQL从库地址都是w.rdc.sae.sina.com.cn:3307和r.rdc.sae.sina.com.cn:3307。直接面对SAE主库、从库地址常量，还有一个好处，就是避免了因为主从同步延迟带来的业务逻辑问题。尽管SAE的内网环境MySQL同步极为迅速，并且一旦发生主从延迟，RDC将会自动将流量导向延迟较小的从库，但仍然不可避免在某些时候的主从延迟影响业务逻辑。比如用户在执行一条Insert后一定要执行Query将之前的Value读出，这种逻辑在读写分离情况下就有可能读不到刚写入的value，所以该场景在执行query时就应该指定仍然向主库执行query操作，这样保证可以读出。

公有云计算数据库平台的难点在于用户之间的隔离性，比如一个用户在一个表结构不合理的数据库上执行性能很低下的SQL语句，很容易拖累别的数据库。在这方面SAE通过RDC来保证，主要

有以下特性。

1. SQL预判。RDC可以提前发现可能损害数据库系统的SQL语句，并执行报警或者拦截。报警是指SQL正常执行，但用户在日志中心会发现该SQL为什么性能不高。拦截是指SQL被RDC屏蔽的，然后用户通过打印mysql_error获取具体拦截原因。正常执行、报警和拦截的区分是通过对用户SQL语句的分析来完成的。

2. 并发执行时间。RDC提出这个概念，来取代传统MySQL的并发连接数限制，原因主要是因为传统的连接数限制较为粗鲁，不能区分用户。换句话说，无法区分“SQL优化的好的用户”和“SQL执行性能低下的用户”。而并发执行时间则很好地解决了这个问题，它突出一个思想：“对于SQL效率高的用户支持更大的并发，而SQL执行效率低的用户则可能不会获得大的并发”，以鼓励用户优化自己的SQL，提高SQL的执行效率，减少对系统的消耗。

举个例子，如果SQL并发执行时间和为10000ms：

- A用户的平均每条SQL消耗100ms，那么A获得的最大并发为100；
- B用户的平均每条SQL消耗1000ms，那么B获得最大并发仅为10。

3. 慢查询配额，RDC通过分析慢查询日志并根据配额进行保护MySQL，一旦发现慢查询超过最高上限的用户数据库，进行限制隔离保护。

KVDB

虽然对用户关系型数据库尤其MySQL往往是最熟悉的，但因为其局限性，在应对数据量巨大的互联网应用时往往有一些不便。比如在SNS中，要存放好友关系，对应的数据结构如下。

- 用户A的好友：用户B；
- 用户A的好友：用户C；
- 用户B的好友：用户D。

从上面的结构可以看到，这类存储的特点是，数据量非常大（N*N），且每条记录字段很简单，且查询需求也不复杂。这类数据就不太适合使用MySQL存储，而适合放在key-value型数据库中，在SAE平台上，就是KVDB。

KVDB支持多种操作接口，包括以下几种：

```
get     按key读取
mget    按key批量读取
pkrget  按key前缀查找
set     更新
delete  删除
stat    查询状态
```

其中值得一提的是前缀查找，这个得益于KVDB底层基于B+树的存储结构。对于使用过Memcached的开发者，再使用KVDB非常简单，因为方式都很类似。

PHP使用的例子：

```
$kv = new SaeKV();
$ret = $kv->init();
if($ret==true)
{
for($i=0;$i<100;$i++)
    $ret = $kv->set($i, 'aaaaaa');
}
```

Java使用的例子：

```
SaeKV kv = new SaeKV();
kv.init();
kv.set("key","value");
Object obj = kv.get("key");
System.out.println(obj);
```

KVDB的最大特性就是无限水平扩展，而无限扩展的背后是其可以实现无缝迁移。分布式数据库系统无论在初始化时怎样通过算法调度调节，都很难避免在运行一段时间后出现不平衡的现象，根本原因就是在于数据库系统本身无法预知用户的使用量和使用方式。不平衡的现象主要是指“容量不平衡”和“负载不平衡”，处理不平衡现象有两个关键环境。

1. 如何判断不平衡，在这点上，KVDB通过基于数学均方差理论的算法从容量和负载两个维度判断不平衡。

2. 如何通过无缝迁移使系统重新达到平衡，在这点上，KVDB通过双线机制来保证迁移时不影响线上服务。

MemcacheX

缓存服务是Web开发最常见的服务，甚至有的个别场景，MySQL都不是必须的，用户直接把他的数据放到Memcached里，然后再定期dump出来。因此，作为面向Web开发者和移动开发者的

PaaS平台，支持Memcached是必须的。

SAE通过自己研发的MemcacheX来提供Memcached服务，MemcacheX的特点是兼容原生Memcached所有接口，并且在性能上做了改进。为了满足PaaS多租户的需求，MemcacheX内部实现了名字空间，不同名字空间的用户的缓存数据不会互相干扰，并且当发生LRU数据置换时，名字空间内部的用户只会置换自己的数据。

开发者在使用MemcacheX时，非常简单。

使用PHP的例子：

```
$mmc=memcache_init();
  if($mmc==false)
    echo "mc init failed\n";
  else
    {
    memcache_set($mmc,"key","value");
    echo memcache_get($mmc,"key");
    }
```

使用Java的例子：

```
SaeMemcache mc = new SaeMemcache();
mc.set("key","value");
Object obj = mc.get("key");
out.println(obj);
```

在Java环境中，因为Memcached并不在J2EE规范中，所以SAE自己实现了SaeMemcache类来满足用户操作MemcacheX的需求，不过仍然保持了简单的风格，尽可能地降低开发者的学习和使用成本。

其他数据存储服务

如前所述，SAE还提供Rank、Counter等一系列数据存储服务，来满足开发者特定场景下的数据存储需求，比如用户通过以下代码（PHP环境）就可以实现自己的一个实时排行榜。

1. 创建排行榜：

```
$sr=new SaeRank();
$ret=$sr->create($name,100);
if($ret===false)
    var_dump($sr->errno(),$sr->errmsg());
```

2. 插入数据，key-value方式，以value为维度做top排名：

```
$ret=$sr->set($name,$key,$value);
if($ret===false)
    var_dump($sr->errno(),$sr->errmsg());
```

3. 获取排名：

```
$ret=$sr->getRank($name,$key);
if($ret===false)
    var_dump($sr->errno(),$sr->errmsg());
```

数据存储服务的选择

SAE提供丰富多彩的数据存储供开发者使用，开发者在使用时也应该了解各个服务的特点，并结合使用场景来使用。MySQL肯定是最通用的场景，最可靠、支持事务、一主多从、有定期冷备，但它也受到容量的限制。另外最重要的是，如果一个PV过亿的应用把自己的数据读写完全压在MySQL也是无法想象的。所以MySQL需要其他数据存储服务来配合，如MemcacheX，轻量级、快速而且高并发，但缺点也很明显——非可持久化，数据可能丢失，而且可能会被自己换出。如果担心这些，那么用户可以选择KVDB，持久化存储而且读写性能只比MemcacheX略低。

当然特殊场景，用户也可以选择Rank或者Counter等服务，这样会使得整体性能更高，比如排行榜服务如果每次都从MySQL进行Order By操作，势必会相当影响性能，而通过Rank服务可以在固定时间获得排名，从而保证实时排行榜。

希望开发者都可以根据自己的需求选择适当的数据存储，这样不仅可以提高用户体验，而且还可以减少不必要的云豆消耗。

从磊

新浪云计算首席架构师，2006年毕业后加入新浪公司，2008年带领技术团队从事云计算方向的技术研发。2009起先后经历了SAE的整个发布过程，目前负责SAE的整体技术架构和研发。

从数据仓库系统对比看Hive发展前景

文 / 杨栋

大数据时代的信息爆炸，使得分布式/并行处理变得如此重要。无论是传统行业，还是新兴行业（特别是互联网行业），日常业务运行所产生的海量用户和服务数据都需要更大的硬件资源来处理。需要并行处理的应用领域主要为网页搜索、广告投放和机器翻译等。从单机应用到集群应用的过渡中，诞生了MapReduce这样的分布式框架，简化了并行程序的开发，提供了水平扩展和容错能力。

虽然MapReduce（Hadoop）的应用非常广泛，但这类框架暴露出来的编程接口仍然比较低级，编写复杂处理程序或Ad-hoc查询仍然十分耗时，并且代码很难复用。目前，Google、Facebook和微软等公司都在底层分布式计算框架之上又提供更高层次的编程模型，将开发者不关心的细节封装起来，提供了更简洁的编程接口。

目前应用最广泛的当属Facebook开源贡献的Hive。Hive是一个基于Hadoop的数据仓库平台，通过Hive，可以方便地进行数据提取转化加载（ETL）的工作。Hive定义了一个类似于SQL的查询语言HQL，能够将用户编写的SQL转化为相应的MapReduce程序。当然，用户也可以自定义Mapper和Reducer来完成复杂的分析工作。从2010年下半年开始，Hive成为Apache顶级项目。

基于MapReduce的Hive具有良好的扩展性和容错性。不过由于MapReduce缺乏结构化数据分析中有价值的特性，以及Hive缺乏对执行计划的充分优化，导致Hive在很多场景下比并行数据仓库慢（在几十台机器的小规模下可能相差更大），Hive的架构如图1所示。

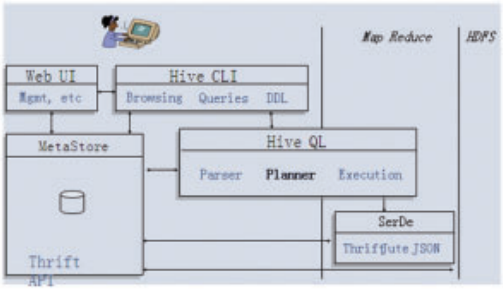


图1 Hive架构图

强大的数据仓库和数据分析平台至少需要具备以下几点特性。

- 灵活的存储引擎
- 高效的执行引擎
- 良好的可扩展性
- 强大的容错机制
- 多样化的可视化

本文将简要阐述Hive是否完全具备了以上几点，以及与传统的并行数据仓库对比优劣如何。

存储引擎

Hive没有自己专门的数据存储格式，也没有为数据建立索引，用户可以非常自由地组织Hive中的表，只要在创建表时告诉Hive数据中的列分隔符和行分隔符，Hive就可以解析数据。Hive的元数据存储在RDBMS中，所有数据都基于HDFS存储。Hive包含Table、External Table、Partition和Bucket等数据模型。

并行数据仓库需要先把数据装载到数据库中，按特定的格式存储，然后才能执行查询。每天需要花费几个小时来将数据导入并行数据库中，而且

随着数据量的增长和新的数据源加入，导入时间会越来越长。导入时大量的写I/O与用户查询的读I/O产生竞争，会导致查询的性能很差。

Hive执行查询前无需导入数据，执行计划直接执行。Hive支持默认的多种文件格式，同时也可以通过实现MapReduce的InputFormat或OutputFormat类，由用户定制格式。因为公司的数据种类很多，存储于不同的数据源系统，可能是MySQL、HDFS或者Hypertable等，很多时候Hive的分析过程会用到各种数据源的数据。当然使用多个存储数据源，除了功能上要能够支持导入/导出之外，如何根据各种存储源的能力和执行流获得最优执行计划也是件麻烦事儿。

强大的数据仓库和数据分析平台需要具备的特性为：灵活的存储引擎、高效的执行引擎、良好的可扩展性、强大的容错机制、多样化的可视化……

执行引擎

并行数据仓库使用优化器。在生成执行计划时，利用元数据信息估算执行流上各个算子要处理的数据量和处理开销，进而选取最优的执行计划。并行数据仓库实现了各种执行算子（Sort、GroupBy、Union和Filter等），它的执行优化器可以灵活地选择这多个算子的不同实现。此外，并行数据仓库还拥有完备的索引机制，包括磁盘布局、缓存管理和I/O管理等多个层面的优化，这些都对查询性能至关重要。而这恰恰是Hive的不足之处。

Hive的编译器负责编译源代码并生成最终的执行计划，包括语法分析、语义分析、目标代码生成，所做的优化并不多。Hive基于MapReduce，Hive的Sort和GroupBy都依赖MapReduce。而MapReduce相当于固化了执行算子，Map的MergeSort必须执行，GroupBy算子也只有一种模式，Reduce的Merge-Sort也必须可选。另外Hive对Join算子的支持也较少。另外，内存拷贝和数据预处理也会影响Hive的执行效率。当然，数据预处理可能会影响数据的导入效率，这需要根据应用特点进行权衡。

扩展性

并行数据仓库可以很好地扩展到几十或上百个节点的集群，并且达到接近线性的加速比。然而，今天的大数据分析需要的可扩展性远远超过这个数量，经常需要达到数百甚至上千节点。目前，几乎没有哪个并行数据仓库运行在这么大规模的集群上，这涉及多个方面的原因。并行数据仓库假设底层集群节点完全同构；并行数据仓库认为节点故障是很少出现的；并行数据仓库设计和实现基于的数据量并未达到PB级或者EB级。

与并行数据仓库不同的是，Hive更加关注水平扩展性。简单来讲，水平扩展性指系统可以通过简单的增加资源来支持更大的数据量和负载。Hive处理的数据量是PB级的，而且每小时每天都在增长，这就使得水平扩展性成为一个非常重要的指标。Hadoop系统的水平扩展性是非常好的，而Hive基于MapReduce框架，因此能够很自然地利用这点。

容错性

Hive有较好的容错性。Hive的执行计划在MapReduce框架上以作业的方式执行，每个作业的中间结果文件写到本地磁盘，最终输出文件写到HDFS文件系统，利用HDFS的多副本机制来保证数据的可靠性，从而达到作业的容错性。如果在作业执行过程中某节点出现故障，那么Hive执行计划基本不会受到影响。因此，基于Hive实现的数据仓库可以部署在由普通机器构建的分布式集群之上。

如果当某个执行计划在并行数据仓库上运行时，某节点发生故障，那么必须重新执行该计划。所以，当集群中的单点故障（可能是磁盘故障等）发生率较高时，并行数据仓库的性能就会下降。在实际生产环境中，假设每个节点故障发生率是0.01%，那么1000个节点的集群中，单点故障发生率则为10%。这个数字并不是耸人听闻的，处理海量数据的I/O密集型应用集群，平均每月的机器故

障率达到1%~10%。当然这些机器可能是2~3万元的普通机型。

可视化

Hive的可视化界面基本属于字符终端，用户的技术水平一般比较高。面向不同的应用和用户，提供个性化的可视化展现，是Hive改进的一个重要方向。个性化的可视化也可以理解为用户群体的分层，例如，图形界面方式提供初级用户，简单语言方式提供中级用户，复杂程序方式提供高级用户。

相关系统

除了Hive，近年来业界系统也诞生了其他各种类型的数据仓库，像Google的Tenzing、Dremel、微软的DryadLINQ、Hadapt的HadoopDB等。Tenzing和DryadLINQ在框架分层上类似于Hive，Dremel除了语言层还实现了计算执行层，而HadoopDB的目标是结合并行数据仓库和Hadoop的优点。

Tenzing的目标是支持Google对数据的Ad-hoc分析，当然之前的分析都是基于传统的关系数据库的。Tenzing在性能方面做了大量优化，包括编译优化以及对MapReduce本身的增强等，这些都使得Tenzing的性能在很多方面接近甚至超过了并行数据仓库。另外，基于LLVM的Tenzing执行引擎可以将单点执行性能提升10倍左右。

DryadLINQ的目标是对并行程序的开发进行抽象，使其和单机开发一样。与Hive不同的是，DryadLINQ并不是一种全新的语言，它将特定语法直接集成到宿主语言（C#等高级语言），充分利用宿主语言的抽象和组合等编程语言机制。与Tenzing类似，DryadLINQ的扩展性和可用性都依赖于底层的执行系统Dryad。

Dremel是Google用于满足交互式查询的系统，主要目标是执行一次性的、结果数据较小的聚合运算，不支持join、update等复杂算子。与Tenzing和DryadLINQ不同，Dremel没有基于MapReduce或Dryad之类的底层计算框架，它实现了一个树状的执行层，这个执行层管理节点的失效或竞争，包括优先级等，与MapReduce的实现机制相似。Dremel的底层存储系统是GFS，基本概念是按列分割数据，把树形结构的数据按列分割。

HadoopDB希望整合并行数据仓库和Hadoop用于大数据分析，这既能保证高性能，又能提供可扩展和高可用能力。HadoopDB的基本思想是在节点上运行并行数据库实例，MapReduce作为这些节点的执行层，尽可能地将执行计划放到并行数据库中完成，利用其已有的优化技术。当然，这会牺牲一部分导入数据的性能，导入数据时会执行一部分表的布局和重组。

总结

作为互联网领域应用最为广泛的开源数据仓库，Hive是份免费的午餐，尤其它在扩展性和容错性方面有强大的优势，其前景被大家一致看好。不过对比传统并行数据仓库，Hive在存储引擎支持、执行引擎高效化以及多样化接口等方面，还有很多工作要做。此外，业界的其他数据仓库，像新兴的Tenzing、DryadLINQ、Dremel或HadoopDB等，都有Hive值得借鉴的地方，Hive原有的实现也还有很大的优化空间。

Hive的诞生带动了Hadoop开源栈系统的进一步发展，也使得很多公司能够从零开始快速搭建数据仓库系统，推动了整个产业链的进步。真心地希望Hive能够继续成长，不断演进，成为全面而强大的通用数据仓库标准。

杨栋

百度分布式高级研发工程师，从事Hypertable、Hadoop及流式计算的研究和开发。

NoSQL持久化

文 / 李刚

关于持久化

对持久化的理解通常是将数据写到断电后不会丢失的设备中，比如磁盘。对于数据库系统来说，持久化是非常重要的功能，它使数据库在经历断电停机后数据不丢失。对于大多数数据库产品来说，这是最基本的功能之一。

NoSQL数据库也不例外，几乎都提供了自己的持久化策略。比如Redis的RDB快照和AOF日志，MongoDB mmap定时fsync以及journaling日志，CouchDB日志即数据的存储模型，等等。本文分析了几种常见的持久化机制，并结合NoSQL产品中的相关应用进行说明。

持久化原理

不同的数据库产品可能选择不同的持久化策略，但从根本上来说，所有的持久化策略都一样，都是将数据写到磁盘（或其他持久存储设备）。不同的持久化策略只是在不同的时机采用不同的方式来完成将数据写到磁盘的工作。

下面我们来看一下将数据写到磁盘的过程是什么样的。

- 调用write(2)系统调用，逻辑上将数据写到磁盘上，这时数据还在系统的内存缓冲区中。
- 操作系统定时或者强制将内存缓冲区中的数据转移给磁盘控制器。数据在磁盘缓冲区中。
- 磁盘控制器将磁盘缓冲区中的数据写到磁盘的物理介质上。数据真正落地。

而在实际环境中，磁盘缓冲区都是关闭的或者为读缓存状态，而不会对写操作进行缓冲。因此，在实现数据库时，需要关心的最重要的事就只有两个：何时调用write(2)写数据到缓冲区，何时通过fsync等一系列调用将缓冲区中的数据写到磁盘。

下面我们分析几种典型的持久化方法。

数据快照

数据快照就是将当前整个数据集按某个时间点进行一次全量保存，将所有数据写入到文件中，然后通过fsync(2)将文件持久化到磁盘。这样在断电后，只要磁盘不损坏，就能通过数据文件将数据库恢复到保存快照时的状态。

这是一种简单粗暴的持久化方式。实现快照的成本很高，通常来说，只在Redis这种内存数据库中比较适用，因为内存数据库的数据全部在内存中，所以遍历全部数据，将全部数据写到磁盘，是其必须要做的事情。在Redis中，实现快照的过程是这样的：首先Redis通过fork(2)操作创建一个子进程，在子进程中遍历数据，将其一条条写到文件中；然后再调用fsync(2)将数据从缓冲区写到磁盘。

Redis每次快照保存都会对全量数据进行遍历并写磁盘，所以当其数据集达到几十个GB时，一次快照可能花费几十分钟时间。

内存映射

内存映射（mmap）其实也可以实现数据快照的功能。这是一种偷懒的做法，因为采用mmap管理数据持久化，就相当于把数据持久化的工作交给操作系统了。mmap的原理就是将虚拟内存空

间与数据文件做一个映射，然后数据库直接在映射的内存空间上进行操作，就相当于是对数据文件进行操作，操作系统会定时将内存中有变更的数据写到对应的磁盘文件上。

与快照相比，它有两个优势：一是采用mmap的方式，数据文件可以远远超过内存大小，理想情况下，只需要内存能装下热数据就行了，因为操作系统的虚拟内存管理机制会将内存自动调度给热数据使用；二是在进行持久化时，需要进行的磁盘I/O减少了。快照方式的磁盘I/O取决于全体数据集的大小，而mmap方式映射的数据文件，在数据库调用fsync(2)将数据写到磁盘时，操作系统会进行判断，只将目前有修改的内存页（脏页）与数据文件进行同步。

正如前面所说，这种方式也可以实现数据快照的功能。不同的是，不是每一次都对所有数据（包括有变化的和没变化的）进行重写到磁盘的操作，而只是对有变化的数据进行重写。

这种方式也可能会导致一些问题，比如在fsync到磁盘的过程中发生断电或者操作系统宕机等问题，可能会导致数据文件中一部分数据是老的，另一部分是新的。这就需要我们再引入其他数据安全性机制来保证了。在后文介绍MongoDB的持久化策略时会说到这一点。

操作日志

操作日志是一种比较老实的持久化方式，它不是将数据进行持久化，而是将数据的操作记录进行持久化。具体的做法是在写操作进行前，先在日志文件中记录本次写操作的修改指令，然后再进行数据的添加、更新或删除操作。这样我们几乎不用关心对数据的修改操作何时真正fsync到磁盘。当数据库出现故障时，我们只需要重复操作日志中的操作，就能够完全重建整个数据库。

使用操作日志有以下两个问题需要注意。

- 第一个问题是操作日志本身的持久化。如果操作日志本身没有保存到磁盘上，那么其在断电后自身都难保，更别说用它来恢复数据了。因此，采用这种方式，操作日志本身的持久化就是数据的持久化。比如在Redis中，AOF就是操作日志的记录，Redis提供了三个级别的日志持久化选项，分别是每次写日志都调用fsync、每秒调用一次fsync和不主动调用fsync。
- 第二个问题是操作日志可能会越来越大，因为任何涉及到数据的写操作，无论是新增、修改还是删除，对于操作日志来说都是一次新增的日志记录操作。所以采用这种方式的话，一定要提供日志切分或合并重写等功能，以保证日志文件不会无限膨胀。比如Redis的rewriteaof功能。

重做日志

重做日志的目的，是通过日志文件的持久化与数据文件的持久化相结合的方式来实现数据的持久化。重做日志与操作日志不同，操作日志只记录进行了哪些操作，它可以独立发挥作用保证数据的安全性。而这里说的重做日志，与操作日志在记录的数据上是不一样的，它不需要保存整个数据库从启动开始的所有操作，只需要保存数据文件在上次fwync后进行的数据变更就行了。

比如，一个将某一条记录的a字段的值从1修改成10的操作，在操作日志中记录的是设置某一条记录的a字段为10，但重做日志记录的是将某一个具体的数据文件中的某一块数据从1修改到10，再将索引文件中的某些块进行相应的修改。有了这个日志，如果数据库在进行数据修改时出现问题，那么我们按这条日志重新执行一次即可。

MongoDB的journal日志就是一种重做日志，当MongoDB在某个时间点异常停机后，就可以通过重新执行journal日志中的更新，将MongoDB恢复到异常停机前的状态。

当然，重做日志与操作日志一样，必须严格控制好日志的持久化操作，才能真正使用日志来对数据进行恢复。但幸运的是，重做日志的记录也是追加式的，因此在日志写入和恢复时的读取上，都是顺序磁盘I/O，不会涉及过多的随机I/O。

持久化选择

Redis

Redis的持久化有两种方式：一种是RDB快照方

式，另一种是AOF操作日志记录方式。

RDB快照就是上面说的数据快照方式。如前所述，RDB快照是通过fork一个子进程的方式，借助fork操作的COW（copy on write）机制，使得数据在内存中可以进行较低成本的快照，然后通过对已生成的内存快照数据全量遍历，再写磁盘生成RDB文件。

RDB文件生成后，不仅可以使Redis支持数据快照的保存。同时也能够用于master-slave模式的初始化过程。

Redis的持久化有RDB快照和AOF操作日志记录两种方式，且持久化机制很灵活，可提供的数据安全性并不比传统数据库低。

下面再说一下AOF操作日志。AOF操作日志是最简单的日志记录，它记录Redis从第一次启动到当前时刻的所有写操作，生成一份完整的操作日志。重新执行AOF中的写操作，我们就能重建整个数据库。

从安全性上讲，AOF的记录方式肯定更好，但AOF日志有上面提到的两个问题：一是数据文件不停增长的问题；二是写操作成本增加的问题。

前面提到过，对数据文件不停增长这个问题，Redis提供rewrite功能对AOF日志进行重写，且默认情况下，AOF会自动重写。如果AOF文件过大，包含了过多的冗余数据，那么它会进行重写，重写过程与RDB文件的生成过程类似，也是通过fork命令的COW特性，生成一个包含现在数据快照的子进程，然后在子进程中遍历数据生成AOF日志。

而对于第二个问题，写操作成本的增加，是无法避免的，世界上没有免费的午餐。不过Redis在这个方面提供了三种不同的配置来控制AOF日志的安全程度：如果你希望每一次操作的AOF日志都安全写到磁盘的话，那么需要在每次写完AOF日志都执行fsync操作确认其安全写到磁盘；如果可以容忍一到两秒的数据丢失的话，那么可以设置每一秒执行一次fsync；如果对安全性要求不高，那么可以设定由操作系统决定fsync执行的时机。当然，这三个选项对性能的影响依次降低，数据安全性也依次降低。虽然如此，但只有Redis出现停机，而操作系统并没有停机的情况下，这三种机制的安全性才一样，因为操作系统迟早会对没有同步到磁盘的缓冲区数据进行同步。

总之，Redis提供了很灵活的持久化机制，使你可以在高性能和高安全性之间进行配置。在开启AOF并且设置总是fsync的情况下，Redis的数据安全性并不比任何传统数据库低。

MongoDB

如前所述，MongoDB的持久化是通过内存映射方式实现的，MongoDB的所有数据文件都是通过内存映射方式进行操作的。这是MongoDB取巧的地方，把困难的缓存工作交给操作系统来处理。由于MongoDB天然采用了这种缓存模式，所以其持久化也就自然而然地通过对mmap的数据文件执行fsync来保证。与Redis相比，MongoDB的fsync时机可配置性稍微弱了一点。Redis可以配置某一段时间有指定数量的操作就生成新的RDB文件，而MongoDB只能指定调用fsync的固定间隔。如果默认为一分钟，那么就会每分钟将内存映射区中的脏页同步到磁盘上一次。当然，也支持手动强制进行fsync操作。同时每次写操作也可以单独设置数据持久化级别。

但如果只使用定时fsync的机制，会有一个问题，就是在操作系统停机时，可能会丢失一段时间的数据。也可能导致一部分数据文件fsync成功了，而另一部分没有fsync成功。如果只是丢失一段时间的数据，通常可以看一下具体的业务能不能够承受。而对于数据文件写花的情况，就需要借助MongoDB的修复功能来做了。也就是对数据文件进行扫描和修复，与用fsck修复文件系统的机制类似。修复是一个很慢的过程，需要对整个数据文件集进行遍历，同时需要大量的空间占用和磁盘I/O。

由于修复的成本太高，MongoDB又推出了journaling日志功能。它相当于一个重做日志，记录了每次写操作修改的所有数据和索引情况。这样在修复时，会通过将journaling日志重演的

方式对MongoDB的数据文件进行恢复。因此你可以看到，实际上一次成功的fsync后，现有的journaling日志就没用了，只需要保存fsync后的重做日志就行了。所以MongoDB在每次实例正常停机时都会将现有的journaling日志清除掉，因为正常停机时，MongoDB会进行一次完整的fsync操作，将数据文件同步到磁盘上。

BigTable模式

许多类BigTable的NoSQL产品，包括Cassandra、HBase等，在数据持久化上都采用了写内存加写日志的方式。这种方式是将写操作生成的新数据记录到内存中，当内存达到一定大小时再将数据整个写到磁盘，这样保证了对磁盘的操作是顺序执行的，这种系统通常有极高的写性能和数据吞吐量。其具体的写操作过程是：当有一个写操作时，数据库会先将写操作记录到日志文件中，然后再写入内存中。这样操作系统宕机时，由于最近的数据都在内存中，所以最近的写操作都丢失了。这时可通过对日志记录进行重演的方式来恢复数据，其原理与上面所说的MongoDB的持久化模型类似。

CouchDB

CouchDB采用了一种很独特的数据存储模式，它的数据写操作是追加型的，不仅包括添加数据的操作，也包括数据的修改和删除操作。因此，CouchDB能够存储一个数据的多个版本，即MVCC模式。由于CouchDB采用了这种方式，其所有写操作都变成了对磁盘的顺序操作，所以不管CouchDB所在的服务器何时停机，CouchDB的数据文件总能保证其完整性。

CouchDB的写操作通常需要两次file:sync调用。如果每次file:sync都直接调用fsync操作的话，性能会比较低，所以CouchDB提供了灵活的选项进行配置。默认情况下是每秒钟进行一次fsync操作。如果在写请求中带了batch=ok选项的话，本次写操作就不会调用fsync同步。如果设定了HTTP头信息X-Couch-Full-Commit: true，那么写操作会在完成fsync后才返回给客户端写成功。

实际上，CouchDB选择追加的数据模型，还有其他一些原因。比如，CouchDB不支持动态查询，所有查询都需要预先建立视图来完成。在查询视图时，CouchDB会查看当前视图的生成时间和当前时间，如果这中间有写操作，那么会将这段时间的写操作增量的进行MapReduce计算，并合并到视图数据中。

在上面的操作中，核心的功能是获取两次访问视图之间更新的数据。而CouchDB这种增量的数据存储模型正好能满足这一需求。我们不用刻意记录每次操作的时间再进行查询，就可以直接截取末尾的一段数据来作为最近一段时间的操作。CouchDB的强大同步机制也得益于其数据追加模式。

总结

本文简单介绍了几种数据库持久化的方式，描述了NoSQL产品在持久化方式上的选择。可以看出，在实现上，不同NoSQL产品选择了不同的持久化方式，这通常是与它们本身的数据存储方式相关的。

同时，相对于传统数据库，NoSQL产品并不是一味地以牺牲安全性来换取高性能或者扩展性的，只是采用了不同的方法，比如尽量将对磁盘的操作顺序化，来减轻在持久化方面的开销。而在你需要时，使用NoSQL也可以达到传统数据库的安全程度。

李刚

爱奇艺高级工程师，NoSQLFan站长，长期关注国内外NoSQL数据库技术。

腾讯的NoSQL应用实践

文 / 吴悦

NoSQL的历史很长，最早可以追朔到Berkeley DB等嵌入式数据库的年代。高速发展的互联网行业对大数据处理的需求日趋强烈，为NoSQL的发展起到了推波助澜的作用。互联网时代的NoSQL，源起于Google为解决大数据的存储和计算而提出的GFS+BigTtable+MapReduce。随后Hadoop（HDFS+HBase+MapReduce）、Hypertable、Memcached、Tokyo Cabinet、Redis、Dynamo和Cassandra等NoSQL产品如雨后春笋般地出现，使得NoSQL技术广泛应用于互联网各个领域。

过去几年中，腾讯在互联网社交平台取得了长足的发展，平台用户基数和应用数等突飞猛进地增长。另外，随着开放的加剧，还有越来越多的第三方选择社交平台开发应用。这些外部条件的变化为技术平台带来了新挑战：除需要为用户提供更强的海量服务外，还需要提供开放的软件基础架构来帮助第三方开发海量服务。

在解决这些问题的实践中，我们总结了很多经验。其中很关键的一点是通过NoSQL技术来构建海量服务的数据层，并通过分析和总结不同的业务场景和技术特点，为各种场景提供更合适的数据层解决方案，如图1所示。

图1 为各种场景提供的数据层解决方案

- 相册、日志等UGC类应用，主要是自我产生数据，他人以浏览为主，其技术特点是读取量巨大，修改量低于读取量一个量级，数据量从几百TB至PB级不等。需要提供SAS、SATA级的TDB和TFS解决方案。
- 农牧场等社交游戏类应用，其核心数据是用户背包数据，互动性很强，其技术特点是巨大读取量与修改量，数据量在百GB级别。需要提供MEM级的TMEM解决方案。
- 信息中心的Feeds类应用，其技术特点是巨大的修改量与读取量，数据量也在几十TB到几百TB不等。需要提供SSD级的TSSD解决方案。

【2006-2008年】因QQ相册而研发TFS、TDB

NoSQL在腾讯的发展历程，需要从2006年TFS的研发开始谈起。研发TFS的目的是在公司内部构建统一的存储平台，为各个BU提供文件系统服务。第一期的重点是能够支持QQ相册的快速发展。当时QQ相册使用传统企业级存储硬件+标准Linux文件系统的老架构，在数百亿图片每天近10亿长尾下载的规模下难以为继。通过分析，老架构主要有以下三个问题。

- 采用FC-SAN等高端企业级存储硬件。这些硬件主要是针对电信、银行等高ARPU值的行业客户而生，价格通常比较高，对盛行免费的互联网企业来说，成本压力大。
- 使用通用的Linux文件系统，不能很好地满足相

册海量小文件的场景下的空间利用率和I/O性能。

■ 元数据与对象数据耦合，扩展性和可维护性较差，单机故障和扩容都是异常繁琐的运维操作。

TFS采用廉价的存储设备，在软件层面使用类似软RAID的技术来满足系统基于不可靠硬件的可靠性要求。将对象数据与元数据分离：对象数据存储采用自研的CHUNK文件系统，inode节点更小；空间分配基于append+delete这种紧凑的管理方式，使得单机最多可以支持数10亿的图片文件；元数据使用MySQL存储。TFS的系统架构如图2所示。

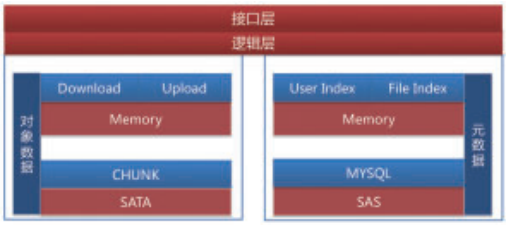


图2 TFS系统架构图

该架构能满足数百亿规模下的QQ相册业务发展。大致在2007年时，QQ相册采用了TFS架构，业务趋于稳定，同时对用户上传放开了限制，因此用户上传浏览的活跃度上升到一个新的量级，用户目录和文件索引等元数据规模突破千亿。然而，在使用MySQL应对如此大规模的元数据的场景下，又暴露出以下一些问题。

■ 索引低效：在QQ相册的场景中上千亿的记录，使用MySQL的B树索引进行索引的存储量消耗在数TB到数十TB。海量索引在无法全内存的情况下会带来I/O的多次访问，一方面增加了单次访问的时延，另一方面降低了磁盘的I/O利用率。

■ 数据搬迁：每天数亿的图片上传导致系统扩容，IDC分布策略导致数据搬迁是常态。使用MySQL的select逐条记录搬迁方式，不同的记录会分散在不同的磁盘偏移，一方面搬迁速度较慢，另一方面由迁移导致的磁盘随机I/O与业务正常访问相互交织在一起，从而影响到在线业务的访问。

■ 系统控制：MySQL主要是针对各种数据通用场景所做的设计与开发，实现较复杂。在使用中遇到性能问题和异常故障时，很难定位原因。对业务系统来说，已经是无法打破的天花板。

针对上面的问题，并结合业务需求，我们在2007年底研发出TDB用以替代MySQL。TDB是一个典型的K-V存储系统，如图3所示。其特点是接口简单、性能高效，具备优秀的扩展能力。

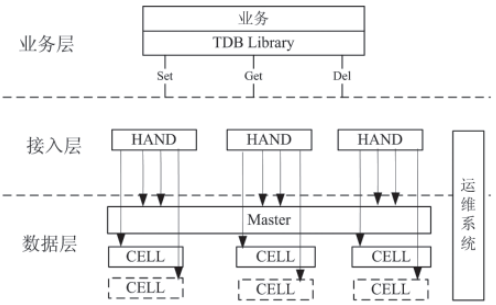


图3 TDB是典型的K-V存储系统

数据层面。索引设计使用Hash，通过KEY直接定位磁盘物理偏移，避免B树设计导致的二次定位磁盘性能开销，解决索引低效的问题。同时采用16MB大磁盘块的设计，使得TDB的数据迁移速度可达网卡性能上限，解决迁移性能问题。另外，系统可控性更强，一方面因为是专用场景，因而可以简化设计，方便定位问题与优化更新；另一方面打通了存储系统到磁盘I/O的控制路径，避免MySQL的系统天花板。

接入层面。为业务提供透明的访问代理，从而实现无缝的水平扩展。由于接口简单，使用非常方便，从2007年底开始，TDB在Qzone、朋友网和群空间等社区应用中逐步取得了广泛应用。

【2009年】社交游戏催生的TMEM

2009年有一款叫农场的游戏，大家应该不会陌生，农场游戏的火爆带动了一批社交游戏应用的兴起。其具备三个典型特点。（1）好友间互动性很强，用户背包数据会被频繁地修改与读取。（2）交叉访问，无明显热点数据。例如，对于传统的应用来说，用户间交互相对较弱，活跃用户数据就是热点数据；而对于社交游戏而言，用户交互性强，通过交叉访问，活跃用户也会频繁地访问和修改非活跃用户数据。（3）放大效果明显。比如，用户每次登录，通常都会到好友的农场去偷菜和捉虫。每个操作都会导致多个用户的多个背包数据的修改。这些行为将导致整个系统中无明显热点数据，用户传统的读缓存+写落地的方

式很难满足这些业务的需求。由于这是庞大的用户基数上的火爆应用，所以单款应用就会在每秒产生数百万次的访问量。这种海量访问不只是对存储层面，同时对网络通信层面提出了更高的性能要求。

很明显，TDB并不能很好地解决这一问题，所以我们在2009年围绕着网络通信和内存持久化两方面，做了大量的设计和论证。2009年底左右推出新的服务TMEM，如图4所示。

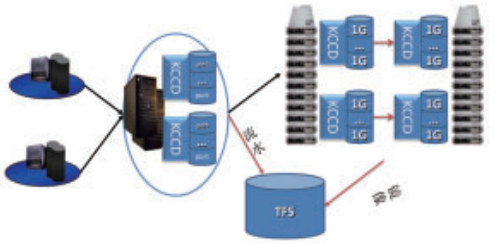


图4 TMEM

TMEM有以下两个核心点。

- 提供内核级KCCD网络通信组件，提升网络通信性能，在网络包量吞吐方面相对于应用层提升了接近1倍的性能。
- 通过操作流水和数据镜像充分利用TFS，解决了内存持久化的问题。

TMEM满足了业务海量访问的需求。但由于内存介质的成本比较高，TMEM在小数据量场景下，性价比较高。但针对中大数据规模的海量访问场景，使用TMEM的成本偏高，而使用SAS介质的TDB，又不能确保I/O性能。比如社区中各类应用产生的Feeds数量为数十TB至数百TB，每天访问量为数十亿至数百亿。

【2010年】顺SSD之势的TSSD

2010年，我们引入了SSD存储介质，开始构建TSSD K-V存储系统（如图5所示）。SSD的特点是：随机读取性能较好，单盘可达数万IOPS，远高于SAS和SATA的数百随机IOPS。容量方面也接近SAS盘，可达数百GB。但SSD也有弊端：寿命有限，随机写入寿命为顺序写入寿命的1/10左右；在随机写入场景下，性能易受干扰，毛刺率较高；受限于物理机制，SSD的存储单元只能先擦除才能写入，并且擦除次数有限，针对NAND芯片，在3000~5000次左右。其中擦除单元是512KB，写入单元是4KB。随机写入的场景，会带来写入放大。

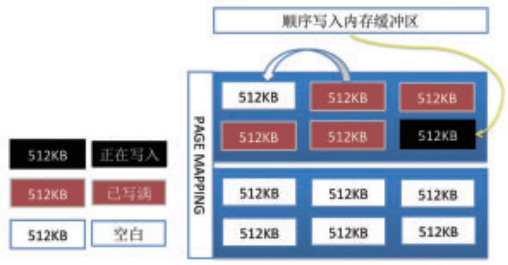


图5 TSSD系统

因此应用SSD存储介质，必须优化随机写入性能。通过构建地址映射，TSSD增加了随机写入内存缓冲区，实现随机转顺序的写入，并通过定期垃圾回收机制，回收垃圾数据。

TSSD系统中，单机可以支持的容量为数TB，性能为随机数万次IOPS。这样基于TSSD使用简单的架构，更少的机器便可支持数十至数百TB的容量，性能为数十万IOPS的Feeds类应用。

NoSQL小结

至此，已构建出基于内存、SSD、SAS和SATA等各类存储介质的存储系统，前面也已提到各类存储系统所对应的使用场景。实际应用中，各种业务场景千变万化，有没有统一的方法来判别和选择合适的存储系统呢？大致在1987年，Jim Gray发表了“五分钟法则”这个观点。简言之，如果一条记录频繁被访问，就应该放到内存里，否则就应该待在硬盘上按需要访问。这个临界点就是五分钟。这看似是个经验公式，隐含的却是硬件性能和成本两方面的因素。大约在1997年时，Jim Gray再次回顾该法则，引入了SSD，验证该法则依然正确。这里不再赘述。

很多情况下需要一种直接根据业务的访问模型，因此使用I/O访问密度，即每GB存储的I/O访问次数，会更为直观。

根据业务I/O访问密度，选择合适的存储介质，就是根据存储介质的I/O访问密度特性和价格来选择性价比最高的存储介质，即找到每种存储介质间I/O

访问密度的临界点。

临界点G（X,Y）.IO per sec per GB = X.IO per sec per GB * Y.price per GB / X. price per GB，X与Y分别表示对比的两种存储介质，且Y.IO per sec per GB 大于X.IO per sec per GB。根据这个公式，可以得到图6中的内容。

存储介质	每GB IOPS区间	系统
SATA	(0,0.15]	TFS
SAS	(0.15,2.32]	TDB
SSD	(2.32,797]	TSSD
DRAM	(797 , ∞]	TMEM

图6 I/O访问密度临界点

怎样理解上面的公式呢？以计算SATA与SAS之间的临界点为例，G(SATA,SAS).IO per sec per GB = SATA.IO per sec per GB* SAS.price per GB/SATA. price per GB = 1/20 * 0.71/0.23 = 0.15 per sec per GB。假设现在有1GB的数据，访问密度是X，X小于2/3，那么使用SAS介质则需要0.71美元，如果选择SATA介质，则需要X/(1/20) * 0.23美元。当X为0.15时，选择SATA和SAS的成本是一样的；当X大于0.15时，则使用SATA的成本比SAS高；否则使用SATA的成本比SAS低。SAS比SSD、SSD比DRAM方法类似。

实际上，对于一款存储介质而言，I/O访问特性与每GB的成本是决定其存在和生命力的关键因素。通常来讲，其I/O访问特性不会有革命性的变化，而每GB的成本是可以控制的，所以厂商总是会不断对容量进行深入优化。

开放的挑战

2010年底开始，随着开放的加剧，越来越多的第三方选择社交平台开发应用。对技术平台的而言，这些外部条件的变化也带来了新挑战：除需要为用户提供更强的海量服务外，同时还需要提供开放的软件基础架构来帮助第三方开发海量服务。成熟的NoSQL架构能否被外部第三方接受的关键点是接口的友好性和兼容性。采用标准化接口，会大大降低外部开发者的使用门槛。因此，在2011年推出了CMEM的NoSQL云存储服务，开发者可直接使用Memcached接口，并即将支持Redis接口，如图7所示。同时也提供了SQL云存储服务，开发者可用MySQL客户端直接访问。后续推出CFS文件系统云存储服务，开发者可使用posix接口访问到TFS。

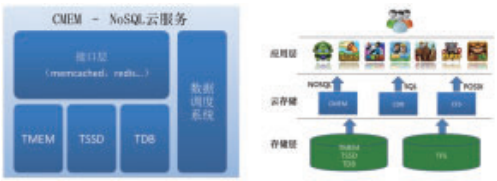


图7 CMEM的·云存储服务

回顾TFS、TDB、TMEM、TSSD、CMEM、CDB、CFS等一系列存储系统过去近6个年头的演进和发展。有以下两点经验可供分享。

■ **建议NoSQL开发者尽量选择构建PaaS服务。**一方面，业务开发者不需关心日常运维，使用更方便，易于接受；另一方面，更容易形成用户需求提出、实现与发布的闭环，从而方便小步快跑，快速迭代出完善的服务。

■ **建议中小业务开发者尽量使用云服务。**通常NoSQL服务所面临的挑战有两个方面：一方面是直观感受到的产品本身；另一方面是服务背后的运营体系。与铁路系统类似，用户直观感受到的火车和铁路只是整个服务中的一部分，是冰山海平面之上的部分；用户感受不到的铁路规划和调度系统等后台运营体系通常是冰山海平面以下的部分，往往需要大量的人力和财力，可能是中小公司难以投入的。而在没有稳固的后台运营体系支撑下，类似动车事故等生产系统故障难以避免，并最终为业务带来不可估量的损失。

从过去几年来看，硬件在变，存储介质的性能和容量在不断提升，并不断有新存储介质的产生；业务在变，随时有可能出现新兴的产品和新的业务体验，对后台系统提出新的挑战和需求。唯一不变的就是拥抱变化，为业务提供更贴切和更优化的存储服务。

吴悦

腾讯架构平台部平台开发中心技术总监，腾讯大讲堂特约讲师，腾讯T4技术专家。先后参与了腾讯分布式文件系统（TFS）、K-V存储、SQL集群和接入网关（TGW）的设计与研发。

剖析同步云存储系统架构

文 / 王奉坤，蒋炜航

随着家庭宽带接入业务的普及和移动互联网设备的兴起，同步云存储服务逐渐进入众多中国互联网用户的视野。这类服务运用互联网数据中心（IDC）的存储和计算能力为用户提供了**云存储服务**，作为本地磁盘存储的备份和补充。此外它还通过提供**同步服务**，使得用户能够在不同的设备和平台上访问到一致的数据。目前常见的同步云存储服务有同步云笔记和同步云网盘两种。本文依照有道云笔记的系统架构，介绍了搭建可靠和高效同步云存储系统的一种思路。

本文的第一部分描述了同步云存储系统的三个设计原则：可扩展&高可靠、支持多版本和模块化。第二部分给出了同步云存储系统架构的示意图，并围绕三个设计原则进一步介绍系统的各个部分。

但请注意，本文描述的架构并不完整代表当前有道云笔记的线上系统，而是基于其架构的一般原理所进行的介绍。

系统架构设计原则

可扩展&高可靠

作为面向个人的云计算服务，同步云存储系统首先必须满足可扩展性（scalability）和高可靠性（reliability）。以有道云笔记为例，假设每位用户实际存储了300MB的数据，而为了提高存储的可靠性，存储系统对这些数据进行了三备份的保存，那么500万规模的用户就需要云存储系统保存4.5PB（4,500,000,000 MB）数据；而5000万规模的用户需要一个能够稳定存储45PB数据的系统架构。此外，当系统不止保存一个版本的数据时（后面会详细描述版本的概念），对系统存储规模的需求也跟着提升。此外，组成数据中心的各种硬件设备都是有可能损坏。例如磁盘的年坏损率大约为2%（一块磁盘在一年内损坏的概率为2%）。假如一块磁盘的存储量为2TB，那么支持45PB数据量的系统至少需要包括22500块磁盘。这样一个规模的云存储系统在每一天都有很大的概率经历一块或者多块磁盘损坏。因此，在设计伊始，兼顾可扩展性（从数十个磁盘扩展到成千上万个磁盘）和高可靠性（在磁盘损坏的情况下仍然正常运转，保持用户数据完整）就成为第一设计原则。

支持多版本

同步云存储系统的另一个重要原则是要能够支持同步协议，允许用户在不同设备上都能读取到一致的数据。除了让用户能够读取最新的数据，同步协议还要处理其他一系列任务，例如将用户在一个设备上做的修改同步到其他设备上；处理串行式修改（逻辑上有先后顺序的修改）；处理非串行式修改（逻辑上无先后顺序的修改，例如用户在两个设备上同时对同一篇笔记进行修改），等等。为了支持同步协议，同步云存储系统的第二个设计原则就是要求它能够支持多版本的存储——保存每一篇笔记最近的几个历史版本。因此，可以看到，云同步系统想解决的问题与SVN（Subversion）解决的问题有些类似，即通过一个集中的存储仓库来提供版本控制。但数据类型和规模的不同使得我们必须重新设计同步云存储系统。

需要说明的是，同步协议本身的设计和实现超出了本文讨论的范围，本文只讨论一个一般化的能够支持多版本数据存储的系统。

模块化

模块化是常见的系统设计原则之一，我们在设计同步云存储系统架构的过程中也积极地采用了这一原则来设计架构。此外，依据可扩展&高可靠原则，大多数模块都能通过添加服务器和模块实例来扩展系统的吞吐能力和提供故障转移（failover）。我们在各个模块之间通过明确定义的分布式RPC接口调用，来确保每一个模块能够按照各自的进度迭代优化。

主要技术点和模块介绍

图1为有道云笔记的总架构图。下面将介绍主要技术点和各功能模块。

可扩展&高可靠——从底到顶的分布式

如前文所述，为了给广大用户提供可靠的海量数据存储云服务，同步云存储系统从最底层的分布式文件系统到最前端的接口协议模块都遵从可扩展&高可靠的设计原则。为此，同步云存储系统采用分布式架构：每个分层都可以通过横向扩展的方式扩大规模；同时，系统中无单点存在，对于突发的硬件故障自动容错。

系统最底层的分布式文件系统（ODFS）是网易有道自行研发的，其实现原理类似于GFS（Google File System）和Hadoop HDFS，是一个主从分布式结构。它由NameNode、Backup NameNode和多个DataNode组成。NameNode上存储的是文件目录树，以及到实际数据Block的索引。Backup NameNode作为NameNode的备份，用以防止NameNode的单点失效。DataNode上存放的是实际的Block数据，在系统运行过程中，可以通过增加DataNode的数量来不断扩展ODFS的存储能力。数据在DataNode间通过三备份冗余的方式备份存储，在个别DataNode故障的情况下，不影响ODFS的数据完整性和服务能力。

在ODFS之上架设有道自主研发的高性能NoSQL分布式数据库——OMAP。OMAP的架构设计与数据组织方式类似于Google BigTable和Hadoop HBase。它同样是主从分布式结构，由Master和

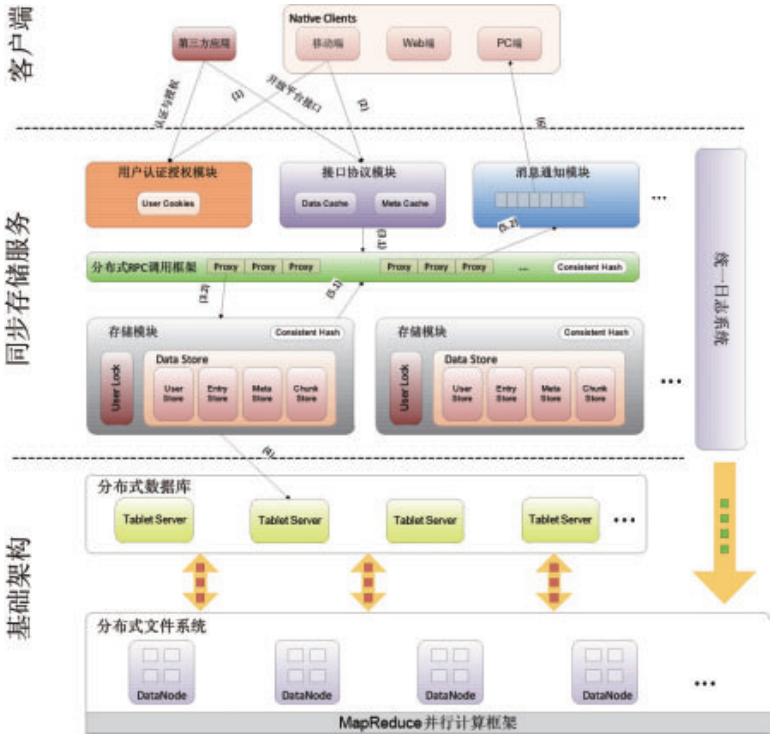


图1 有道云笔记总架构图

分片服务器（TabletServer）组成，通过KeyRange方式对表格（Table）进行分片（Tablet），每个分片使用预写日志（Write Ahead Log）、写内存（WriteBuffer）、持久排序字符表文件（SSTable）等方式组织数据。作为一个可扩展的系统，增加分片服务器的数量就可以扩展其负载能力。OMAP同时还是一个高可靠性系统，Master的短暂故障不会影响整个系统的服务，并且可以通过迅速重启Backup Master的方式消除Master的单点；局部分片服务器的故障可能会短暂影响部分数据的读写，OMAP会自动迅速地恢复服务。

与Hadoop HBase相比，OMAP还针对实际需求做了一些优化：提供单纯的KeyValue式存储，取消对Column Family和Version的支持，换取更高的读写效率；因表格而异的Cache策略，用LRU算法缓存读写最为频繁的数据，提高Cache命中率；对于读取效率要求特别高的表格，可以通过配置将持久存储的数据全部加载到内存或者SSD中，大大提高数据读取速度；采用JNI方式管理WriteBuffer、Cache和BloomFilter等数据结构占用的内存，降低Java FullGC的执行压力，进一步提高效率。正因为这些优化，OMAP的读效率是

HBase的两倍。

在OMAP之上是有道云笔记的分布式存储服务模块——DSS。分布式存储服务模块维护数据的存储逻辑，但其本身并不存放数据，而是基于OMAP提供的存储服务存放数据。DSS采用同构的分布式架构，通过自协调、推选Master的方式实现分布式逻辑。Master负责生成Consistent Hash，并通过开源软件ZooKeeper将Consistent Hash同步到各个存储服务模块实例，各个实例通过Consistent Hash来确定自己所服务的用户。这样就保证了在同一时刻，一个用户的所有终端都会由某一个特定的存储服务模块实例服务。因此，存储服务模块实例使用内存锁就可以对单个用户有效加锁，将单个用户的访问在服务器端串行化，有效避免了用户并发访问带来的操作原子性问题。

可扩展&高可靠——增量备份

除了通过分布式文件系统（ODFS）对用户的数据进行三备份存储，我们还在OMAP中添加了数据增量备份功能。OMAP通过表数据快照（Table Snapshot）和预写日志（WALog）的方式实现数据的增量备份。一旦OMAP本身的数据出现错误或者损坏，可以通过MapReduce并行计算框架来读取预写日志（WALog），并在表数据快照（Table Snapshot）上重放操作，以恢复数据。通过这种方法，可以将数据恢复至表数据快照（Table Snapshot）创建的时刻与当前时刻之间的任意时间点。如图2所示，如果用户数据在t时刻发生严重错误，通过重放t时刻之前的预写日志（WALog），可以将数据恢复到错误点之前。然后，将新生成的表数据快照复制回到OMAP，消弭t时刻错误对用户

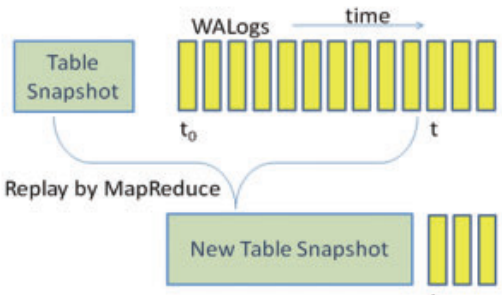


图2 基于快照和预写日志的增量备份

数据的影响。

支持多版本

有道云笔记在分布式存储服务模块层使用版本的概念组织用户数据。由于同一个数据可能被多个终端修改，所以为了防止出现类似于数据库中的脏读（Dirty Read）的现象，服务器端应该对同一个用户的所有修改操作进行串行化，即任意时刻服务器只响应来自一个终端的修改请求。

在传统关系型数据库中，我们可以通过事务来保证这样的操作，但在分布式系统中，要维持一个高并发的分布式全局锁，其代价是非常高的。这里可以考虑使用另一种方式，首先通过用户分发机制保证，在任意时刻，有且仅有一台服务器为同一用户提供服务，而此时只需要在每台服务器上分别维护一个用户锁即可保证同一个用户的所有修改都是串行化的。

另一方面，为了防止一个终端在提交修改请求时，服务器端的数据已经被另外一个终端所修改，即幻读（Phantom Read）现象，我们又引入了版本的概念。与SVN类似，每个用户的全部数据都可以看作是一个代码库，拥有唯一的版本，在修改请求到来时，请求所携带的版本号必须与服务器端该用户的版本号一致，修改操作才会提交运行，否则终端将收到版本冲突的错误。

各个终端需要自行处理这种情况，例如可先更新至服务器当前的最新版本，进行数据合并，再重新提交修改请求。通过这种乐观锁机制，各个终端在提交修改请求前都能够意识到当前服务器端用户数据的状态，防止用户在无意识的状态下覆盖了之前所做的修改。

由于有了版本信息，用户每次所做的修改数据也都可以与版本信息相结合，这样在服务器后台看来每次更新操作也可以统一为一次插入操作。从功能上讲，通过这种方式，用户之前的历史版本也会得以保留。如果有需要，用户可以回滚至之前的历史记录。而从效率上讲，由于分布式NoSQL数据库处理插入的性能通常要好于删除或更新操作，这种处理方式也在一定程度上提高了系统的吞吐性能。而真正的数据删除操作则可以

放在定期的空间回收中来完成（通常可以选在系统闲时进行），可以根据不同用户的配置来采用不同的回收策略。

模块化

除了前面介绍的ODFS、OMAP和DSS模块，同步云存储系统的主要模块还包括用户认证授权模块、接口协议模块、消息通知模块和统一日志系统模块。这些模块都可以通过在多台服务器上运行多个实例来提供更强的服务能力。它们之间通过使用分布式RPC相互调用，屏蔽分布式多服务器的细节。下面将简要介绍各功能划分模块。

分布式 RPC

分布式RPC调用框架是一个具有分布式分发功能的RPC工具，能够屏蔽分布式RPC服务的细节，为服务使用者提供一个简单透明的RPC调用方式。

用户认证授权模块

该模块处理所有与用户认证和授权相关的事务，各个客户端，包括第三方应用程序，需先通过该模块进行用户的登录与授权，登录成功便可以取得相应用户的cookie，之后各个客户端便可以利用cookie访问接口协议模块来进行数据的同步。该模块同时提供了RPC接口供其他服务模块来判断用户传入的cookie是否有效。

接口协议模块

该模块提供了用户数据同步的全部API，是各个客户端访问的主要模块。该模块通过RPC接口来访问存储模块中的数据，出于性能的考虑，模块也提供了缓存。由于版本信息的存在，即使该模块分布于多台服务器上也不会有缓存脏数据的出现。另一方面，由于不同客户端的计算能力与本地存储能力不同，接口协议模块也提供了不同层次的访问接口。

消息通知模块

该模块处理用户数据更新时的消息推送。当用户数据在任何一个终端上发生修改并同步后，其他在线的客户端都会收到服务器推送的数据更新消息并触发一次自动同步，使得所有在线的客户端数据始终保持同步状态。

统一日志系统模块

由于不同的服务模块甚至同一模块都会分布在不同的服务器上，所以同一用户的操作也会被分配到不同的服务器上执行。统一日志系统可以将不同服务器上的日志汇总到一起进行分析，并通过SQL进行查询，从而可以按照时间查看某个用户的所有日志，进而定位问题。

总结

本文以有道云笔记的系统架构为原型，介绍了一种设计同步云存储系统的思路。在网易有道基础架构团队的多年打造下，ODFS和OMAP不但在有道云笔记中发挥了巨大作用，还为有道搜索、有道词典等系列产品提供了基础服务。本文介绍的其他部分是有道云笔记团队专门为同步云存储场景而设计的。从2011年6月有道云笔记对外服务至今，这个架构很好地满足了用户对存储和同步的需求，承受住了指数级增长的系统压力的考验。因此，我们希望通过分享我们所理解的同步云存储系统架构，为提高中国云计算领域的技术实力抛砖引玉。

王奉坤

毕业于北京航空航天大学，就职于网易有道，高级软件工程师。参与有道云笔记服务器端分布式架构设计、软件开发、NoSQL数据库的开发等工作。

蒋炜航

University of Illinois at Urbana-Champaign计算机博士，网易有道云笔记团队负责人。20世纪80年代开始编程，曾就职于HP Lab和NetApp，在硅谷参与过创业。

淘宝网“双12”背后的技术故事

文 / 陈国成

对于淘宝网而言，2011年的“双12”是一个交易里程碑，是淘宝网最大的购物狂欢。疯狂涌入的千百万买家，为淘宝网带来了43.8亿的历史单天最高成交金额。2011年12月12日零点过后第一分钟内，瞬间涌入的买家人数高达270万，一小时之内，成交额达到4.75亿，成交278万笔，创造了历史峰值。

这是对淘宝网系统和团队的全方位考验。所谓的“双12”技术，是淘宝在高性能、高并发、安全、稳定、容灾、数据分析等领域多年的积累：从业务平台到基础平台，从硬件到软件，从网络到存储，从容灾到监控，从沟通到协作，从意识到流程，以及无数次的经验教训等，而非一朝一夕之功。要完整描述“双12”等类似的大促行为对淘宝网所有系统、所有部门带来的挑战，以及应对策略，是件很困难的事情，而且也不容易聚焦，因此本文侧重描述核心的在线交易系统面临的主要挑战，以及系统的应对方法。

大促的挑战

高压力。所有业务数字，最终都会转化成对系统的压力数据，如TPS、QPS等。以核心交易系统中的商品平台为例，上述交易数据意味着该平台需要支撑的最高瞬间压力达到每秒近25万次的读请求（QPS），每秒近万次的事务写请求（TPS）；全天需要支撑的总请求达到几十亿次以上，并且系统负载始终保持在可接受的范围之内。

持续高压下的稳定性。大促当天，系统压力持续在高位运行。如果平时的压力测试方法存在遗漏，系统中存在内存泄漏等隐患，那么此时必定对系统的稳定性造成很大影响。

突发事件处理。包括紧急的脏数据处理和数据批处理等。

沟通协作。要保证大促成功进行，需要协调数十个团队一起努力。

结构与技术特点

在介绍关键技术之前，先花稍许篇幅交待一下淘宝网核心交易系统的结构及各系统的主要职责。

图1是隐藏了很多技术细节的核心交易系统的结构图。从宏观层面讲，核心交易系统主要有以下几个特点。

依赖

淘宝网的整体架构是从2009年开始奠定下来的。那一年淘宝对核心系统进行了大规模重构，根据业务、技术等特点，将核心业务系统拆分成前台、中心系统、后台和数据层四个层次，改变了以前的两层结构，即应用层和数据层。淘宝现在的应用系统（前后台系统加上中心应用等）有几百个，核心交易系统也有几十个。

图1 核心交易系统结构图

数据层有持久化（DB）、缓存化（tair、与memcache类似）和搜索化（实时/非实时搜索）等多种数据源，提供核心商品、交易、用户和店铺等数据的存储和检索。中心应用通过封装对数据层的访问，对外提供API，供各个前后台系统访问。前后台系统对淘宝网用户（买家、卖家）和淘宝网小二提供GUI，供用户访问和执行业务操作。系统间的通信，主要通过淘宝的高性能并发框架HSF（同步）和消息中间件（异步）进行，部分通信通过JSON（HTTP）等方式进行。

这种结构使得交易核心流程变得清晰、简单和独立，但同时也使得数据的访问路径变长，任何一个交易流程中的系统的稳定性，都会对上游的系统的稳定性造成影响。与两层架构下的情景一样，如果DB不稳定，毫无疑问访问（依赖）DB的系统也会脆弱无比。

为了保证交易核心流程的稳定，必须对整个平台进行准确的依赖关系建模，包括以下几方面。

- 依赖关系：例如A依赖于几个系统，对这些系统的依赖是强（同步调用），亦或是弱（异步、间接调用等）。
- 依赖方式：A对所依赖的系统，采用何种协议和通信方式。
- 依赖数量：A每天及高峰时对所依赖系统的调用次数。
- 响应时间：A对每一个依赖的系统调用的响应时间。

这个过程花费了很长时间，并且是一个持续完善、优化的过程。但只有严格的依赖关系模型，才能使得系统可控性成为可能。如果无法梳理该关系模型，则数据访问路径的稳定性和可控性根本没法保证；极有可能虽然花费了很长的时间对某个系统进行优化，但在大促时却被短板系统搞得灰头土脸。

应用分级与隔离

主要是中心应用的分级。中心应用非常重要，它们通过服务化接口或客户端直连数据源等方式，为前后台系统提供核心数据（商品、订单、店铺和用户等）的访问。所有的前后台系统，无论是否为交易核心流程的系统，都会通过中心应用进行核心数据的访问，这就会带来一个极大的挑战：流量分配。

例如，中心应用A每天对外提供的服务能力是20亿次，有50个前后台（其中有交易核心流程，同时也有非核心流程的应用系统）的系统依赖于A。如果50个前后台系统访问的均为同一组集群，那么假设某天一个后台系统的需求暴增，超出了A的服务范围，则极有可能使交易核心流程应用系统的稳定性和访问需要受到影响，进而影响交易的稳定性。解决方法是对中心应用进行集群分级（在通信框架层面），提供不同稳定级别的多组集群，供不同的前后台系统调用，保障为安全和稳定要求高的系统提供服务，将不稳定的系统隔离开。

读写数据的解决之道

就淘宝网一类的网站而言，对数据的访问，永远是读远远大于写。下面大致介绍了淘宝对数据读写访问的主要形式，以及技术上的处理方式。

数据读。对核心数据（商品、订单等）的读需求主要有以下两种。

- 主键访问：以商品详情页面为例，要根据商品主键访问商品的数据信息。

主键访问方式的访问量一般比较大，往往会超出数据库的服务能力；另外，由于B/S架构的特点，对数据的读访问通常并不要求达到完全实时。因此，解决这种读请求的办法是使用集中式缓存：将要访问的数据全部装入缓存中（tair，与memcache类似）。

用缓存方式解决主键访问，需要先解决好数据一致性、数据延时、数据预热等问题。

- 模糊匹配：以搜索为例，卖家管理商品时以模糊匹配方式检索商品。

我们采用的主要解决方案是倒排索引，如阿里集团研发的isearch，或开源的Lucene；个别情况下也可以采用BDB实现。

数据写。写数据，只能实时地操作数据持久层

（数据库）。对于极个别的例外场景，如秒杀或类似OceanBase一类的NoSQL存储引擎，会首先将数据的写请求，在内存中进行多次合并，将数据写请求合并成单次对数据持久层的操作，从而大大减少写数据对数据持久层造成的压力。

缓存

缓存分为集中式和单机式两种。集中式如tair，主要缓存商品等数据。对于这一类缓存，采用合适的序列化解决方案至关重要。对于商品，淘宝选择的序列化方案是protobuf，数据压缩算法选用lzf；与Hessia+gzip相比，数据容量上节省了41%的空间，性能提高了16%。

淘宝有一些静态的字典数据，需要缓存到前后台Web系统的内存空间中，以便在需要时能高速访问，这就是单机式的缓存，典型的例子是类目属性数据。对于单机式缓存，有一套完善的监控和推送（定点、定时、定向）的配套系统非常必要。另外单机式缓存有时占用的内存空间会很庞大，甚至会达到800MB。除了对每个DO进行极致的瘦身之外，淘宝采用内存结合文件的形式解决了这个问题：将数据索引（B+树）和特别热点的数据停留在内存中，将其他数据交换到本地磁盘，从而将对内存的占用率减少了50%，缓解了内存的困境。

持久化存储

在2010年和2011年，基于成本、性能和水平扩展等方面的考量，淘宝对核心数据（商品、用户、订单等）的存储层（DB）进行了重大改造，全部迁移到了MySQL+PC Server+Fusionio架构的DB层，替代了原IBM小型机+Oracle+EMC高端存储的DB层。新的持久化存储，带来了更低的成本（降低了80%左右）和更高的性能（TPS提升5倍，QPS提升10倍以上）。

图2 商品数据库的能力模型：容量

图3 商品数据库的能力模型：流量变化趋势

此外，淘宝在存储和检索一体化方面还有一个非常优秀的NoSQL存储引擎——OceanBase，它很好地满足了收藏夹海量记录存储和检索的需要。

完全定制、个性化的架构

业务决定了技术的架构。由于面向的业务有诸多特殊性，所以淘宝的各个系统很难有一种“通用”的架构和技术来满足所有业务系统的技术需要。以持久化存储为例，有一个场景是，需要存储商品的变更记录。商品变更记录每一天会新增1.2亿条左右，要存储全年的变更即为400亿行；每天读的请求数在10万次以下，QPS在个位数级别。在这样的场景下，如果采用传统的关系型数据库，其成本无疑相当巨大。因此定制了一个非常微型的DB，索引与数据文件分离，存储分成增量和存储两部分，分别在本地磁盘和HDFS（静态查找表）上存储，并在I/O上做了一些优化。基于该架构系统，仅使用两台服务器，就可以支撑以上数据的存储和检索需要。

关键技术及其实现

大促对系统所提出的更高要求，促使我们需要在以下方面做得更加深入。

可精确度量的系统能力

淘宝的所有核心交易系统，都建立了非常精确的能力模型，即能支撑的最大TPS、QPS和PV等。数据的精确性和全面性至关重要，对于作出决策具有决定性的作用：集群是否需要扩容，幅度多少，在最坏的情况下系统的反应如何，一旦崩溃如何快速恢复等。为了获得精确的系统能力数据，我们除了采取测试环境持续的压力测试外，还在线上环境进行持续的真实压力测试。交易核心路径上的每一个系统（包括数据库），都需要经过持续不断的测试，得出最精确的能力模型数据，如图2和图3所示。

通常情况下，将系统拖到崩溃边缘的，并不是平均负载，而是波峰负载，即瞬间最大并发访问。因此对该数据的估计需要特别精确。同时所有的核心交易系统都需要遵循余量策略，保证至少3倍以上的余量。

性能优化

性能优化是一种很常态的工作。包括应用程序端（前端和后端）的优化、CDN优化、图片优化、cache优化和业务逻辑简化等。

监控与容灾

为了保证稳定性，淘宝有“持续稳定性平台”，时刻对各个系统进行监控：系统负载/响应、JVM健康状态和实时报警等。系统的任何问题，如程序异常、响应超时达到阈值、线程堵塞和I/O异常等，均可以监控到并及时报警。监控的准确和及时，对于线上故障的处理速度和避免影响面的扩大等具有决定性的影响。

监控是一种持续行为，即便在非大促时期，仍然有专门的团队履行着这一职责。同时，核心交易系统中各个子系统和新系统的架构，都需要具备可监控性。平台的稳定始终是业务发展中权重最高的因素之一。

数据分析

阿里巴巴拥有着规模非常庞大，并且成熟、稳定的云计算平台，基于该平台，我们能够对十几亿的商品进行热度等分析。提前预知热点商品分布情况等，以便及时发现对系统的潜在隐患。

降级及开关

在最坏的情况下，一旦我们对系统可能面临的压力估计不准确，系统将很有可能会到达崩溃线，此时应急处理措施非常关键。除了在硬件上加后备集群之外，各个交易核心系统均在主流程中增加了各种开关，以便在需要时降级某个模块。

降级，指的是页面/服务端可以不展示某个模块内容或不进行某个模块的计算，从而节省运算资源或节省数据库开销等。将该模块内容屏蔽的手段称为开关，可理解为若页面上的一小段逻辑为true，则忽略某个模块的运算和展示。

可降级的模块的识别原则，主要依照该模块对于买家下单的影响程度。若影响较小，则可以进行降级；而部分模块非常重要，降级之后会对买家的交易造成影响，则不可以降级。

流控

流控是保护系统的最后一道防线，即当被调用的系统的压力超出了该系统的能力时，暂时“下线”对该系统的调用；一旦压力恢复正常，系统自动恢复原调用。流控的逻辑实现是在调用端进行的。典型的流控示例是对数据库的保护，如数据库能承受的每秒并发访问是1000，而假设并发访问该数据库的应用服务器有100台，则通过在每一台应用服务器上增加计数器等方式，记录对该数据库的并发访问总数；一旦达到10个，则新的请求将被block住，如图4所示。

流控需要处理好对业务的影响，尽量避免影响到核心的交易流程。

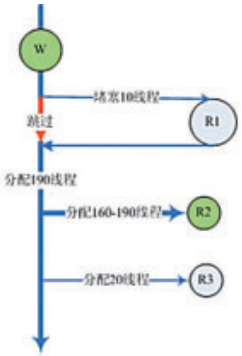


图4 流控流程

大促总结

对于工程师而言，大促的最高境界是没有大促战斗：无需特殊优化，无需人工监控，与平时工作日一样平常。我们也在不断努力，为所有在这个平台上交易的用户，提供最稳定、最可靠和最友好的电子商务平台。

陈国成

淘宝网高级技术专家。熟悉复杂的电子商务商品业务体系，以及商品核心系统的技术架构及实现。有着超广泛的求知欲，除技术和业务外，对管理饶有兴趣。

云计算与大数据挖掘

文 / 何清

大数据的基本概念

根据维基百科的定义，大数据（Big Data）是一个用于数据集的术语，指数据大小超出了常用软件在运行时间内可以承受的收集、管理和处理能力。大数据的其他英文表述方式还有海量数据（Massive Data）、大规模数据（Large Scale Data）、庞大的数据（Enormous Data），以及巨量数据（Huge data）。

大数据的界限随着技术进步而不断增大，它的规模是一个不断演化的指标，目前范围是从数十TB到十几PB级别。

大数据的出现源于我们生活在不断密集使用数据的技术社会之中。世界上有46亿移动话单，10亿~20亿人访问互联网。目前，人们正处在一个“无处不网、无时不网、人人上网、时时在线”的时代。图灵奖获得者吉姆·格雷（Jim Gray）曾说，网络环境下每18个月产生的数据量等于过去几千年的数据量之和。目前互联网的数据具有海量增长、用户广泛、动态变化等特征。与数据和信息交互的人比以往任何时候都多。根据Cisco预计，到2013年，互联网流量将达到每年667EB。

大数据存储方式经历了人工管理阶段、文件系统阶段、数据库系统阶段、分布式文件系统阶段和NoSQL阶段。随着数据量增长，越来越多人开始关注NoSQL、HBase、BigTable和Hive。

2001年Gartner发布报告指出，数据增长带来3个方面的挑战和机遇，即增加容积（数据的数量）、速度（数据吞吐速度）和多样化（数据类型和来源范围）。

大数据带来的技术挑战包括：收集、存储、搜索、共享、信息和可视化。大数据需要特别的技术在可容忍的时间里有效地处理大量的数据。用于大数据的技术包括大规模并行技术MMP（Massively Parallel Processing）数据库、数据挖掘网格技术、分布式文件系统、分布式数据库、云计算平台、互联网和可扩展存储系统。这些都与云计算技术密切相关。

大数据挖掘云服务

深度处理与挖掘要求

数据挖掘在人工智能领域，习惯上又称为数据库中的知识发现（Knowledge Discovery in Database，KDD），也有人把数据挖掘视为数据库中知识发现过程的一个基本步骤。数据挖掘过程由以下3个阶段组成：数据准备（数据预处理）、数据挖掘和结果表达。

数据挖掘是通过分析每个数据，从大量信息中寻找其规律的技术，主要有数据准备、规律寻找和规律表示3个步骤。数据准备是从相关的数据源中选取所需的数据并整合成用于数据挖掘的数据集；规律寻找是用某种方法将数据集所含的规律找出来；规律表示是尽可能以用户可理解的方式（如可视化）将找出的规律表示出来。

数据挖掘的任务有关联分析、聚类分析、分类分析、异常分析、特异群组分析和演变分析等。大数据为数据挖掘技术的发展创造了空间，基于大

数据的挖掘技术历来受到重视，追求全量数据挖掘的高效算法是数据挖掘研究者面临的最大挑战。云计算技术的引入为大数据挖掘提供了计算环境和能力。

数据挖掘面临的主要问题是，要处理的数据是海量、高维多模态、多类多格式的。因此采用的数据挖掘算法往往是高复杂度的，而数据挖掘的结果往往又要求高度精确，能直接用于具体对象的判定和操作。数据挖掘面临的往往是NP问题，迫切需要近似线性，或者尽量低阶的多项式复杂度算法，需要高超的将递归改为循环（迭代）的技巧和高效的并行策略。

高效并行大数据挖掘算法

MapReduce并行编程模型具有强大的处理大规模数据的能力，因而是海量数据挖掘的理想编程平台。数据挖掘算法通常需要遍历训练数据获得相关的统计信息，用于求解或优化模型参数。为了实现大数据上的数据挖掘，为数众多的分布式并行数据的挖掘算法在近些年被提出。

在大规模数据上进行频繁的数据访问需要耗费大量运算时间。为了提高算法效率，斯坦福大学的Chu T. Chu提出了一种适用于大量机器学习算法的通用并行编程方法。

通过对经典的机器学习算法进行分析可以发现，算法学习过程中的运算都能转化为若干在训练数据集上的求和操作。求和操作可以独立地在不同数据子集上进行，因此很容易在MapReduce编程平台上实现并行化执行。将大规模数据集分割为若干子集分配给多个Mapper节点，继而在Mapper节点上分别执行各种求和操作得到中间结果，最后通过Reduce节点将求和结果合并，实现学习算法的并行执行。

在这个框架下，Chu T. Chu实现了10种经典的数据挖掘算法，包括线性回归、朴素贝叶斯、神经网络、主成分分析和支持向量机等，相关成果在NIPS 2006会议上发表。

Colby Ranger提出了一个基于MapReduce的应用程序编程接口Phoenix，支持多核和多处理器系统环境下的并行程序设计。Phoenix能够进行缓存管理、错误恢复和并发管理。他们使用Phoenix实现了K-Means、主成分分析和线性回归3种数据挖掘算法。

Gillick对单程学习（Single-pass）、迭代学习（Iterative Learning）和基于查询的学习（Query-based Learning）3类机器学习算法在MapReduce框架下的性能分别做了评测。他们对并行学习算法涉及到的如何在计算节点之间共享数据、如何处理分布式存储数据等问题进行了研究。

Mahout是APS（Apache Software Foundation）旗下的一个开源数据挖掘项目，通过使用Apache Hadoop库，可以实现大规模数据上的并行数据挖掘，包括分类、聚类、频繁模式挖掘、回归、降维、协同过滤与推荐、LDA等算法，目前已经发布了6个版本。

数据挖掘面临的往往是NP问题，迫切需要近似线性，或者尽量低阶的多项式复杂度算法，需要高超的递归改为循环的技巧和高效并行的策略。

Qing He等几个人提出了基于MapReduce的并行增量ESVM分类算法、完全并行的决策树算法、高效并行CLARANS聚类、基于KD树的并行异常发现算法。

Kanishka Bhaduri整理了一个十分详尽的并行数据挖掘算法文献目录，包含了关联规则学习、分类、聚类、流数据挖掘4大类分布式数据挖掘算法，同时还包括分布式系统、隐私保护等相关的研究工作。

基于云计算的大数据挖掘服务

云计算除了给用户提供通用的并行编程模型和大规模数据处理能力外，另一个重要的特点是为用户提供开放的计算服务平台。在数据挖掘方向，现在也有一系列的系统被开发出来，面向公众提供数据挖掘服务云计算服务。

为什么要用云计算的方式来对数据进行挖掘？从需求来讲，所要处理的对象数据是海量的，我

们以往期望使用高性能计算设备来完成。但事实上，前文已经提到，互联网的数据增长速度很快，数据挖掘的任务远比搜索更复杂。这导致了我们在挖掘过程当中需要有更好的开发和应用环境。这种情况下，云计算能提供强大的计算能力。

而从外部特征来看，基于云计算的数据挖掘，对于没有条件自己搭建数据挖掘平台的中小企业很有吸引力，因为在云平台上能够以租用的方式进行挖掘，处理成本被大大降低。在并行条件下，服务提供方还可以利用原有设备，方便地增加节点，使容错性更强。

PDMiner是由中国科学院计算技术研究所2008年底开发的基于Hadoop的并行分布式数据挖掘平台，该系统现在已经用于中国移动通信企业TB级实际数据的挖掘项目。基于云计算平台和MapReduce编程模式，PDMiner中包含了各种并行数据预处理操作及并行数据挖掘算法，这些并行数据挖掘算法包括关联规则算法，分类算法以及聚类算法等。提供了一系列并行挖掘算法和ETL操作组件，PDMiner开发的ETL算法绝大多数达到了线性加速比，同时具有很好的容错性。

另外，PDMiner系统还开放了灵活的接口，方便集成新的ETL算法和数据挖掘算法，这种开放式架构可以使用户将算法组件经过简单配置方便地封装加载到系统中。如果用户自己没有精力做挖掘算法的基础研究，那么PDMiner则提供了一个很好的技术平台，使用户不用投入大量的研发资源进行基础算法研究，只需要关注在自己的业务流程和用户访问模型的研发，使用PDMiner简单拖拽即可完成分析挖掘。另外，由于PDMiner有很好的可扩展性，应用开发商可基于它开发应用，到用户的实际系统中进行部署。

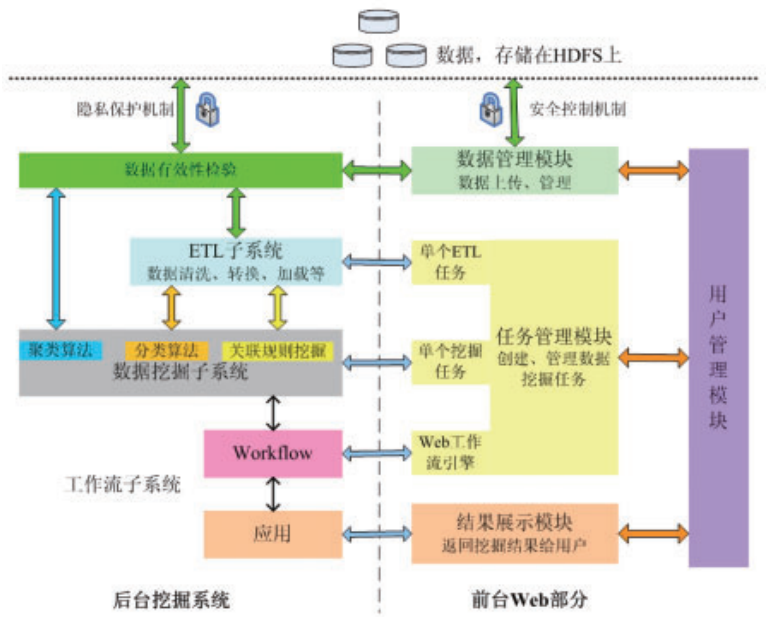


图1 并行数据挖掘软件系统COMS体系结构

典型应用

由于挖掘对象是中国移动的电信数据，其规模已经超过了许多商业软件所应对的环境，但得益于系统从设计之初就考虑了分布式环境，使用PDMiner进行数据挖掘的效果比较理想。

典型的例子是，以消耗计算资源较为突出的数据挖掘为例，相对于小型机、中型机上的专用数据挖掘软件，基于云计算的并行处理的数据挖掘系统，具有更快的处理速度和更方便的计算可调性，尤其在经营分析或处理大量客户信息时。中国移动通信集团一个中等规模的省公司拥有大约1000万用户，每年产生的CDR（呼叫话单）数据量大约在12TB~16TB。即使对于一个非常简单的业务目标进行数据挖掘，经过预处理后，算法也需要处理大约10GB的数据。而一个省公司的网管数据量更是可达到每天1TB量级。随着应用需求愈加复杂及变化多样，数据挖掘应用向其IT支撑平台提出了更高的计算及存储能力要求。

另外，数据挖掘应用也逐步提出实时性要求，拥有及时的商业策略才能快速占领市场。但传统数据挖掘系统运行于Unix小型机的集中平台，受到很多限制。目前，以一个聚类应用为例，现有的商用数据挖掘系统仅能支持100万用户1个月内数据的知识发现，这距离实际的要求相差甚远。传统方式处理1TB的数据挖掘需要8个小时，而使用16个节点的云计算平台则只需要40分钟，且成本仅为传统方式的四分之一。

云计算使得海量数据的准实时处理成为可能，从而带动了经营分析和决策优化，开启了精准营销和基于用户偏好、用户期望的服务模式的未来。

在PDMiner基础上，中国科学院计算技术研究所还开发了数据挖掘云服务平台COMS。该平台的体系结构如图1所示。

COMS系统由前台Web部分、后台数据挖掘系统以及硬件资源管理系统3部分组成，数据通过硬件资源管理系统存储在HDFS上。

硬件资源管理系统

CMOS对本地的硬件资源进行管理，记录各计算节点的状态，将可用的计算资源通过Web交互界面呈现给用户，分配用户请求的计算资源，管理用户的计算资源。例如用户申请的节点宕机之后进行重新分配。还可以回收计算资源，即当某一用户的挖掘任务完成后回收其申请的计算资源。

前台Web部分的结构和功能

CMOS用户管理模块的功能包括用户的创建、修改、身份确定、用户授权、权限收回、用户日志的记录及维护等。用户根据该模块定义的身份及权限同数据管理模块、任务管理模块及结果展示模块进行交互。

任务管理模块提供用户执行挖掘任务的功能需求，包括任务的创建、修改、执行、停止等。用户在该模块能够使用的功能由用户管理模块定义的用户权限决定。该模块可与后台的ETL子系统和数据挖掘子系统交互，执行单个ETL或者数据挖掘任务。

用户结果展示模块能够使用的功能也由用户管理模块定义的权限决定。该模块和后台系统中工作流子系统的应用部分交互，获得挖掘结果，并以用户定义的形式呈现给用户。

后台挖掘系统的结构和功能

数据校验模块提供用户数据校验的功能需求。根据其所选择的任务类型，对用户提交的数据进行校验。

该模块在读取用户提交的数据时，需要有相应的隐私保护机制，对用户提交的数据进行保密，防止外泄。

并行ETL子系统提供数据预处理（清洗、转换、加载）的功能需求，包括主成分分析（PCA）、统计、抽样、缺值处理、行内去重、标准化、离散化、区间化、数据整合、属性约减、数值替换、属性交换、属性删除、属性添加、标识符添加等操作。该模块为数据挖掘子系统提供符合要求的数据，也可与前台的任务管理模块交互，让用户完成单一的ETL任务。

并行数据挖掘子系统提供并行数据挖掘的核心，包括数十个算法用于关联分析、聚类分析、分类分析、异常分析等核心数据挖掘功能。

工作流子系统提供并行数据挖掘中多挖掘任务一站式挖掘的功能需求。例如，先对数据中的缺失值进行缺值处理（替换为固定值、常用值或直接置为空），再对数据中的连续型数据进行离散化，最后用处理好的数据训练一棵决策树。

继中国移动之后，国外商业智能领域的各大公司也陆续推出了面向企业的大规模数据挖掘服务，例如IBM、Oracle等公司都拥有自己的基于云计算的数据挖掘平台，这些平台各有特色，各自依托原有的技术优势，例如与自己的硬件设备紧密结合，面向特定的客户群体和行业领域开展应用。

总之，云计算为大数据处理和挖掘提供了计算技术和计算环境，大数据推动了云存储、云挖掘与云服务的发展。

何清

中科院计算所研究员，博士生导师。中国计算机学会高级会员，人工智能与模式识别专业委员会委员。中国人工智能学会副秘书长，中国电子学会云计算专家委员会委员。

自己想办法

有关工程现状的几点反思

文 / 周爱民

从过程、方法和工具构成的工程理论，到人本思想为基础的敏捷实务，作者指出种种工程理论中最基础的组织因素不过是一事一职，并提出了以项目、团队与产品三种角色及其视角构成的具体化的工程思维方向。

如厕的方法

古人是很讲程序、方法与规则的，这其中就包括如厕这件事。而且，这件事的程序性相当重要，器具制式和用法也有定规，以至需要佛祖释迦牟尼来详加解释。这件事情记在佛经里，要求如厕者常自备“厕筹（木片）”；如果没有，那也不能拭在墙上、厕板上，也不能用石头、青草、土块、软木之类代替；而应该现找木、竹、苇，临时用作厕筹。不单如此，佛祖还解释了厕筹的长短制式，以及用完了之后不能用弹、振、甩等方法来弄干净，也不能跟干净的厕筹混放在一起。这便是佛祖他老人家传下来的“上厕用厕筹法”。

佛经原文大抵便是如此记载。所以那个时代的僧侣们，大概是要随身备厕筹一枚的。倘若一时忘掉，那便只能折了木、竹、苇来替用。再如果这些东西都没有，可能就必须蹲在原地，等着其他人来，或是借用，或是求人家去取。但顺着这个思路想下去，便发现：如果僧侣在四野无人的地方遇上了这个问题，既没有人又没有替代物，那么岂不是要活活地“蹲到死”？

几千年了，从没听说过有佛教徒这样蹲到死的。即便这个范围放得宽大一些，我也没听说过有哪个人是会这样蹲到死的。

没有人会机械地遵循某种“如厕方法”所约定的过程，一旦有“标准方法”所未顾及的、不太周全的情况发生，我们总会自己想办法解决问题。从来没听过会有人自己提了裤子出来，做好了厕筹再蹲回去完成程序的。

不具体的工程

而在我们的工程中，“过程论”的专家总会要求你这么做。同样的，这样的专家在面临“没有厕纸”这样的境况时，宁可触墙而死也不会上厕所，因为没有任何可用的过程、方法与工具。

佛祖其实是个好人，他老人家介绍自己的最佳如厕实践：厕筹最短也不要短过四指，要不用着不方便；不能把染着污秽的厕筹甩来甩去，要不沾到别人，起了纠纷，打得头破血流；用过的厕筹不应乱放，要不会传染疾病或影响别人使用……佛祖的如厕方法的确是很好的，有着远比别人丰富的如厕经验，且可行性相当强。

但无论如何，你得先有厕筹。因此工具论也就大

行其道，以至于南唐后主——就是写“春花秋月何时了”的那位——也要亲自去削竹片木片来做，并且还要放在脸上试用，若有锐利不适的地方，再加以修正。所以厕筹这个东西，也是可以做得相当考究的，例如在外面拿香囊之类来包裹，既款款有型又不失雅趣，这样的事情在东晋也是有的。

先有了方法，而后，其细节的步骤被规则化了，就变成了过程论的论调；其细节所依赖的那些东西，就构成了工具论的基础。如此一来，“过程+方法+工具”，就构成了我们将一件事情工程化的全部。

当一件事情的做法被工程化之后，事就不再是具体的事了，成了一类工程——或是统一工程，或是敏捷工程，总之都失去了“具体”二字。

不再具体，也不再立时可用，也不再（仅仅）是一个法子。

传统的工程

软件工程原本是只讨论“过程+方法+工具”的，这是纯粹工程学的视角[1]。这其中即使有一些看起来与人相关的话题，也不会被过深地讨论。例如过程中显然涉及到人，但不会因此而讨论到“过程中的产品环节，是否可以由开发人员来兼任”这样的问题。

事实上，在“纯粹的”软件工程的讨论中，过程的各个环节都是由一些“角色”来推动的，这些角色如何存在于一个具体组织的部门中，其职业定位、能力结构等，都并不算一个严格意义上的工程话题。

但这样的“纯粹”显然遇到了挑战。当我们试图将工程完全理论化之后，它就必然面临实践的困境。“人”的问题在这方面首当其冲。“人月神话”并不是一个“人/月”的度量理论那样简单——事实上《人月神话》那本书很少讨论工程度量的问题。“人月神话”在工程上的重要性在于，它严肃地提出了“人是不是工程学问题的一部分”这样的话题。这本书的答案将这个话题引向了一个极其巨大的迷局，即如果需要更多的人来实现工程，则应付由人带来的复杂性，可能将超过工程本身的技术复杂性。《人月神话》一方面有先见地对技术复杂性提出了一些可靠、可行的方案，另一方面也悲观地认为由人带来的复杂性必然导致“巴比伦塔”最终的倒掉。“没有银弹”的论证过程将所有的焦点集中于：通过（较小规模的）程序实现的过程，无助于求解（包含大规模工程在内的、普遍含义的）根本任务。

作者Brooks把以下三个问题从《人月神话》的讨论中摒除出去。

- 项目必是有始终的。
- 哪怕是一个不成功的产品，也需要交付。
- 团队是人的问题，并不是事的问题。

《人月神话》将以上三个问题的前题隐设为：项目需要承担大量的附加责任，产品必须是尽善尽美的，以及人是私利的。而这正体现了“工程学”的学术本质，即如果可以的话，应当基于工程来解决一切问题，因而它首先必须能解决一切问题。

在这些年的实践中，这是我所见的对“一件事”具有最大伤害的但又貌似义正辞严的、跨越行业与组织的普遍观点了。我之所以用这样复杂的定语来说明这一“普遍观点”，在于它确实那样“自然而然”地存在着，就像每个行业都受了同一个神灵的启迪一般。事实正是如此。

任何一个有道德、正义而又富有职业责任感的产品经理都会跳出来，说“这个产品没做好是我们的责任”，然后历数种种设计细节，再加以细化，并立时加入需求库；任何一个有同样品质的项目经理也会跳出来，说“这个项目没做好是因为我们对流程的贯彻不够”，然后开始制定更详细的管理措施；然后开发人员也跳出来，说出超过200个的技术改进点……

以某类角色为中心的工程

而每个人都将问题揽在自己头上，这难道不是一项好的品质吗？如果是，那么我又怎么会用一种调侃的语气来说出上面的假设？

这样的假设，在于警示我们一个忘掉的事实：之

所以这是一个系统，并不在于一人一己或者一个角色的过失，也不取决于某一个角色的单方面的努力。这些“积极、冲动而又充满道德责任感”的人，事实上是无助于找到系统问题的答案的。现实中，我们可以找到许多具有这样性质的工程、产品与过程模型。在这样的模型中，（提出模型者所代表的）中心角色总是认为自己应当能够且也必须能够去承担更多，因而也就更重要，其他所有角色都应该围着自己打转转。

然而这样一来，每个角色都站在一个自己认为“更合适”的角度来看我们在做的事情，而全然不管那件事情本身为何。例如，开发人员可能认为自己的某项技术发明有着相当“巨大”的前景，于是期望有一个团队来配合自己将它实现为产品。至少在他看来，由技术来推动产品和市场是相当伟大的成就。但一旦他提的这个想法如同扔进了泥坑一样丝毫得不到反馈，他就会立即开始着手“发明”下一个技术与想法了。

你会发现，他根本不需要对产品和市场负任何责任。但如果他只管提出想法，而不需要负责任，又怎么可能站在别的角色上去思考问题呢？

所以再读《软件工程》这本教材时[2]，就只会看到一些“管理的技术”了。然而我从不指望把“管理”当成技术手段和工程方法的人能真正地做好项目。这里，我指的是眼前的具体项目。

敏捷工程

如上所讨论的，在发现“人的问题”是一个关键因素之后，传统工程就将它作为一个学术话题，用一些角色、规格与方法来限制这些人的行为，试图将管理变成一个“技术所需的条件”以及“可以基于技术手段来改善的群体工件”——“人件”这个名词其实就翻译得很好。《人件》本质上也是基于这样的立论来讨论工程话题的一本书。简单地说，就是总有一些方法可以让管理成为工程的一部分。

但敏捷工程却在“人的问题”上探索另一个“解”。关于敏捷工程，我从来都是把它与传统工程“对立”起来讨论的。但在根本事实上，它们是不对立的。也就是说，它与传统工程在“关注人的问题”这一点上并不对立，但在二者的求解方法上大相径庭。

敏捷工程直接将问题指向“人的私利”与“合作”之间的矛盾，即人们究竟为什么要合作。敏捷工程无比强调团队内部的合作，并将这一合作设定为一种“道德需要”。由于有了这个前提，不合作者将被套上道德枷锁，或被拷问，或被驱逐。总而言之，敏捷团队必将因此变成一个（尽可能地）倾向合作的团队。因为这一核心前设的不同，在四条敏捷宣言中就有两条涉及合作（另外两条则是针对产品定义的），而在十二条敏捷原则中，则有超过半数都是在强调团队成员的道德认可，而非强调某种技术方法。

从这个角度上来说，敏捷工程的成功之处在于形成益于合作的团队氛围，以及在道德与行为准则上实现“强制合作”。

具体化的工程

无论敏捷工程还是传统工程，都无法为你正在做的这件事提供直接答案，得靠自己努力。

一个具体化的工程，涉及三个方面的问题：人与团队、产品与用户、项目与方向。如果要将这些东西“系统化地”组织在一起，必然是需要将它视为“一件事”的，这也是它被“立项”的目的。如果这一前提不存在，则产品只是用户调研、产品设计或产品线上的一个规划，而可以将团队简单地视作“一群人”。当要做一件事时，这三个方面的东西才发生关系，也才交织在一起成为“问题”。

要开始做“这件事”，会有不少前提条件。例如作为一个项目经理，你或许要求“先做三个月技术培训，再开始开发”。但事实上这些都算不上“前提”，因为也可以不这么做就匆匆上马——只要你有机会在团队行进中调整大家的步伐。唯一能作为“开始这件事”的充要条件的，事实上只有一个决策性的判断：我们是否需要用工程化的方法，来实现一个产品目标。这里涉及到以下三个问题。

- **是软件产品吗**？如果这不是一个需要“软件生产”的产品，那么可能不需要“软件开发”这一

过程，固而也就不需要软件工程。例如市场部门需要了解客户反馈，那么它既可以实现为一个在线聊天的客服机器人，也可以实现为包含大量职员的客服部门。而后者显然是不需要“软件生产”的。

- **需要开发吗**？即使是一个软件产品，获得它的方法也可能不是“软件开发方法”。例如上述的“在线聊天客服机器人”，如果我们将它作为一个产品外包，那么“外包”这件事对于我们来说就是一个供需管理，而不是“工程化”的软件开发活动。
- **需要工程化方法吗**？一个研发性软件产品的开发过程，可能不是“工程化”的。例如，我们真的要在公司内立项来做上面这个产品，但只分配了一个开发人员或一个临时小组，产出的产品也不用于市场销售，对时间的控制也可以放得很宽松，而且最为重要的事情是，假定这个项目是基于技术研究的，因而当它失败了也不会有什么负担，那么我们便不需要考虑“是不是需要一个工程化的软件开发方法”。

当决策判断发生了，当“应作为一个工程化的软件产品，来予以立项”成为事实时[3]，产品、团队与项目三者就必须立即成为“具有相同目标”的一个合作群体。这个相同目标，就是“做一件事”；这个相同目标中的道德原则也只有一个，就是“做完”。

成事之本，完事之心

我知道有一个做了整整五年的项目。项目持续了这么久，是什么在支持着这个团队一直保持着追求成功的激情呢？就这个问题，一个普通成员的回答很发人深省：

“在我们的头脑里，完成一个项目是最重要的事情，即使我们不喜欢这里，也不会让项目流产。我们宁愿先完成项目，然后在第二天辞职。”

这个团队就是Windows NT 4.0的开发团队。在读《观止：微软创建NT和未来的夺命狂奔》这本书的过程中，这名普通的、甚至没有在这本书中留下名字的成员，说出了让我如此印象深刻的话。这样的普通成员，正是这个团队整体的职业精神的缩影。

回顾《观止》，我们是无法孤立地站到团队、项目与产品三个视角之任一来思考的。从根本上，具体工程最重要的话题，就是这三个视角必须是统一的、系统的，以及面向具体的“一件事”的。

注释

1. 事实上还会讨论到“目标”问题。但在传统工程中的“工程目标”并不是确指，而是使用“需求”来指代的一系列技术方法。传统工程基于“需求”而非基于“具体目标”，并且——与此相关的——它与架构的“面向问题”也是相互背离的。因此传统工程中的、工程师视角下的思考，对“架构师”这一角色在团队中的融入帮助并不大。

2. 早些时候的《软件工程》教材是不讲团队、管理等内容，但后来它们都逐渐地加入了相应内容。老实说，这些书（如《软件工程：实践者的研究方法》与《软件工程（Ian Sommerville著）》等）中有关于人的讨论都是隔靴搔痒，无济于事的。这些经典的、学术化的工程方法仍是以自己为中心的，它们试图把管理学、组织学或社会学“拿来一用”，其思想离“看到一个项目自身的系统性”还相当之远。

3. 事实上三种情况的反例都是可以“立项”的，譬如组织建设或人力规划的项目，企业资产购置与管理的项目，实验性项目。它们作为项目是满足“时间、质量与成本”这样的项目要素的设定的，但都不是工程化的。

周爱民

资深软件工程师、架构师。有十余年软件开发、项目管理、团队建设的经验，曾任盛大网络平台架构师、支付宝业务架构师等职。著有《JavaScr ipt语言精髓与编程实践》等书。

责任编辑：杨爽（yangshuang@csdn.net）

程序员的创新修炼

文 / 许正华

以系统的方法来理解创新思维的基本方面有助于了解持续创新的内在规律。本文作者根据多年的工作体验和思考，展现出了一个循序渐进的创新思考模型，并结合实例进行了深入的阐释和分析。

关于创新

对程序员来说，“创新”是一个永恒的话题。它给世人的感觉是既简单又玄妙。说它简单是因为创新似乎不需要严格的外在条件，说它玄妙是因为我们很难说清楚创新产生的过程，所有的创新几乎都是不可复制的。

【实例分析】前几天听到同事在抱怨没有足够的硬件资源做伸缩性测试，经过进一步询问，我了解到：产品中有一个部分需要多任务（进程）执行；任务进程原本是由C++实现的，为了调用第三方Java SDK远程访问WebService，每个进程都加载了一个Java虚拟机。经过测试发现使用了此SDK的JVM占用了大量的系统资源，最终大大降低了系统的可伸缩性。面对同样的场景，人们的反应会有很大的差异：有些人会抱怨几声了事；有些人会设法寻找更为强大的机器；还有人会尝试从软件架构的角度改变现状。例如，把通过Java SDK远程访问WebService的逻辑封装为本地代理服务，它可以将相关业务逻辑的重用从组件级提升为应用级（如图1所示）。没错！这就是创新。本文将放弃任何对结果的讨论，因为创新不应以成败论英雄。

如果说这些微创新不但是每个程序员都可以做、而且必须要做的工作，那么我们是否已经准备好了呢？这方面的系统论述向来是比较少的，水平

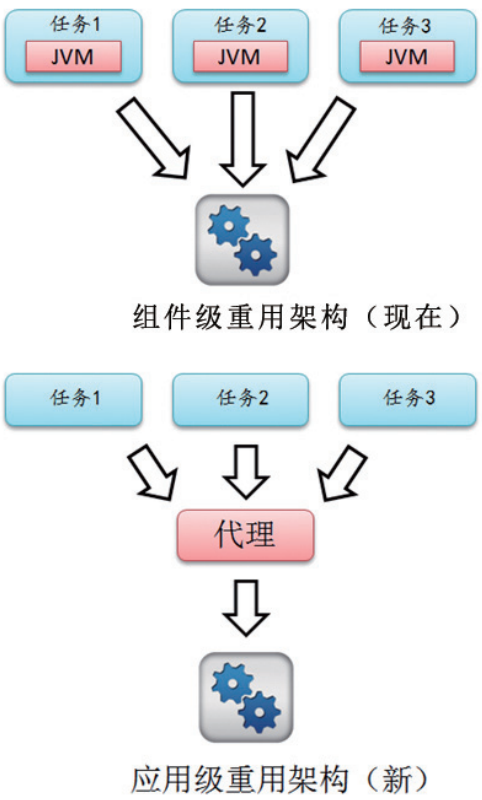


图1　重用架构升级

思维理论的缔造者和倡导者德·波诺教授在《比知识还多》一书中提出了一系列可用于提升思维创造力的工具（例如，通过随机输入突破现有思维模式、挑战概念、跳出主要观点、定义问题和剔除错误等）是这当中杰出的代表。在我看来这些工具并不是独立的，它们相互影响、共同发挥

作用。接下来就走近它们，以探其真面目。

批判性思维

批判性思维是创新过程最显著的特征（也就是剔除错误），就让我们先从它开始吧。

在程序员江湖闯荡久了，就会深知创新的艰辛。不求一鸣惊人、但求独善其身是程序员队伍中一种比较具有代表性的思潮。我们真正能做到独善其身吗？试想如果现有代码的可维护性很差，任何人都很难在有限的时间内写出高质量代码，最终很可能落得个同流合污。

有时我们也会一不小心走到另一个极端：如果留意，我们总能在公司的茶水间里听到员工关于公司缺乏产品创新的恶评。批评是有益的，但我们更需要批评过后的冷静思考和实际行动。创新无疑需要批判性思维，但前提是我们能将“批判”与“批判性思维”区别开：批判的目的只是为了达到对某个事物的否定；而批判性思维则是通过否定来寻找更好的做事方法。单纯的批判是消极的；而批判性思维则要积极得多。

问题的 定义

批判性思维是建立在问题定义基础上的，因为“没有问题就无所谓批判”。

什么是问题

什么是问题？期望与现实的落差就是问题。现实是容易觉察的，但界定外界期望就难得多了。人们倾向于把外界的输入作为建立合理预期的唯一标准，而不是独立形成自身对预期的认识。例如，开发人员总乐于将测试人员在缺陷描述中记录的期望行为和结果作为实际的用户需求，并据此来修改代码。

【实例分析】某产品中定义了“策略”的概念，并提供了相应的编辑界面。在策略编辑界面（如图2所示）中，允许用户对原有策略进行保存或另存为一个新策略。测试人员曾经报过一个用户体验方面的缺陷：用户在选择“另存为”功能时，很

图2 策略编辑界面

可能会遇到系统提示的失败报告，原因只是忘记了修改策略名称。

我们可以顺着测试人员的思路将问题定义为“如何让用户避免令人沮丧的失败报告”。进一步思考，不难发现策略编辑界面还有其他问题，比如，用户原本希望创建一个新策略，却错误地点击了“保存”按钮。因此，也可以将问题定义为“如何通过合理的功能布局，避免不能预期的用户行为”。最后，我们决定提供一个“策略拷贝”的新功能，并将“另存为”按钮从策略编辑界面中移除。

在定义问题时，我们需要注意以下几点。

- 一个现象可能隐藏着多个问题。
- 区分核心问题与非核心问题；可以存在多个核心问题。
- 问题可以在不同抽象层次上定义（从用户体验的角度来看，可分为表现层、框架层、结构层、范围层和战略层）。

随着被灌输信息的增多，人们所受到的来自外界的影响和约束不断增强。一个人对现实的接受程度越高，问题形成的难度就越大。

外界是不完美的

对于软件研发团队，定义问题的能力直接影响其需求分析的质量，进而最终决定了产品的质量。一般认为软件工程师参与到需求分析与产品设计的机会并不是很多，他们接收来自用户或者产品经理的需求规格，并且努力把功能列表转变为可以运行的程序。不幸的是，现实中软件生产过程要复杂得多：在开始阶段，写下来的需求只是冰山一角；未写下来的需求才是沉浸在水下的巨大

冰块；即使写下来的、被用户所确认的需求规格也不能确保最终一定能够让用户满意。

《用户故事与敏捷方法》在论述用户故事与需求规格、用例等传统需求分析工具的区别时，特别强调用户故事的目的不是记录客户与开发团队之间的协议或契约，而是充当着用户具体需求对话的占位符。

因此，我们不能把产品经理的输入等同于用户需求。在现实中，开发团队总是容易认为项目早期形成的用户故事就是对用户需求的描述，因此大部分人都放弃了对问题进行深入且持续的思考。敏捷实际上是想告诉我们：外界是不完美的，所有相关人员（产品经理、开发者、测试者、用户、文档编写者等）只有通过不断的沟通才能加深对用户需求以及产品策略的理解，并在讨论的过程中达成一致。如果我们不能在这个方面放弃传统的惯性思维，我甚至怀疑所谓的敏捷是否还具有生命力。

外界的不完美导致我们必须关注自身定义问题能力的培养。首先，通过独立的思考形成自己对一个事物的预期，而不是唯命是从；其次，通过各种可能的手段来确认这个预期是否合理，并进行必要的修正；最后，对比现实与预期形成对问题的认识。当这种思维模式成为你的一个思考习惯时，问题就会不断地从脑海中涌现出来。

概念的建立

建立概念模型的意义

合理预期是建立在合理的概念模型基础上的。只有明确了概念，才知道未来的路该往哪里走，才知道什么时候需要坚持原则、什么时候需要像现实妥协。只有建立起概念模型，你才有可能进一步思考现实与存在于你脑海中的概念是否一致。

每一个软件设计问题的背后都隐藏着一个或简单或复杂的概念系统。很多时候，软件工程师并没有意识与能力去主动地发掘这些概念，这是导致软件设计不确定性的重要原因之一。因此，对概念的驾驭能力就成为软件架构师最为倚重的基本能力之一。

忽视概念的历史根源

对概念的忽视有其深刻的历史原因，如下所述。

炒作概念。不知从什么时候开始，“概念”常常与一个动词相关联，它就是“炒作”。一个新产品往往借助炒作热门概念来提升自身的价值，“概念”渐渐变成了假、大、空的代名词。这与概念本身的功能完全背道而驰——概念原本是为了让一个或一类事物能够更准确地被大众以一致的方式所理解。说句公道话：“之所以盛行炒作概念之风，不是因为今天概念太多了，恰恰是因为现如今人们缺乏对概念的理解。”这一切大概发生在过去的十年里（十年前的我大约刚刚完成学业、步入社会）。

只摸石头不过河。我们不妨把时间推得再久远一些：在我念书的年代里，对概念的深入研究变成了教条与迂腐的代名词，因为对很多人来说概念意味着陈旧，意味着一成不变。概念确实表现出一定程度的稳定性，这种稳定性源于其抽象的本质。但我们切不能将这种稳定性绝对化。概念同样也会像这个世界上的绝大多数物质一样，随着时间的流逝发生变化。这种可变性源于其主观的本质。没错！概念并不是像人们想象的那样客观。人们只是试图通过自身的努力将自己对概念的理解无限趋近于客观。

“实践是检验真理的唯一标准”——这是几乎每一个中国人都熟知的一句话。我从不怀疑其合理性，但放在上述社会环境中，难免容易让人迷失：既然是“检验”，我们就应该首先澄清待检验对象。换句话说，我们应先形成对真理的假说，然后不断地用实践结果增加其可信程度，亦或反之。现实中，人们渐渐丧失了事先形成对真理认知的耐心，这种不耐烦也许是源于人们对于可能出现的自我否定的恐惧，这直接导致实践最终仅仅成为一些人收集数据的工具。

我曾与一些软件研发人员讨论有关软件性能测试的问题。我发现他们并不能清楚地说出某个测试的目的。当我进一步询问测试结果能否帮助我们得出一些结论时，他们就变得含糊其辞。在我看来：单纯为收集数据而进行的测试是毫无意义的，这就好比只惦记摸石头而忘了过河才应是原

本的目的。

缺少点科学精神。曾经的诺贝尔奖得主杨振宁在对比中华文化与近代科学时，有如下精彩总结：中国传统　文化所追求的“理”与近代科学所追求的“自然规律”方向上一致。但它们在求“理”的方法上是不同的。传统中国文化主要依赖归纳法求理，这里面没有逻辑与推演；而近代科学除了使用归纳法以外，更依赖演绎法。

显然，对于建立复杂的科学体系而言，演绎法更高效。这就不难解释：为什么自然科学在近现代中国更多是以舶来品的形象出现？演绎法需要逻辑，而逻辑建立在概念的基础之上。如果缺少这方面的先天基因，那么就更需要后天努力了。

乐观精神

没有精神层面的支持，创新是无从谈起的，因为创新实在是一项艰巨的工作：概念的建立不是一蹴而就的，需要反复的推敲、对比、自我否定；问题的定义更是自我意识游走在理想与现实之间，尝试各种问题边界的可能，不断地进行权衡；人生贵在坚持批判性思维，而不沦落于简单的批评与抱怨。所有这些行动都是很困难的，因为它们会产生巨大的认知摩擦。由此可见，对于一个致力于创新的程序员，能够建立并保持一个积极的人生态度才是最值得庆幸的事情。

当我请求同事们进行自我评价时，大多数人都会为自己打上乐观的标签，因为他们把乐观与快乐、自我满足等概念混为一谈。一般人通过努力可以让自己维持一个相对快乐的心境，但要做到乐观就很难了。快乐是一中状态，而乐观是一种态度；快乐是相对短暂的，而乐观是相对持久的；快乐既可以源自内心，也可能是源自环境，而乐观是一定是由内而外的。

创新思考模型

我们可以将创新模型简单地归纳为以下四个核心要素。

- 创新需要挑战现状，即批判性思维（要素）。
- 批判性思维需要准确的问题定义（要素）。
- 问题的定义有赖于概念系统的建立（要素）。
- 在初始阶段，概念模型的正确与否是次要的，更为重要的是要有一个概念模型，然后这个概念模型可以随着认识的深入而不断演化。谁在幕后推动认识的发展？答案还是批判性思维。批判性思维有助于我们修正原有概念的不足或创建全新的概念，在思维的各个阶段都需要乐观的人生态度的支持（要素）。

回到开篇的实例中，让我们看看这些要素是如何发挥作用的：我们不能接受过高的资源占用这个事实，即使是在硬件变得越来越廉价的今天，滥用系统资源永远是值得我们去批判的。问题的关键在于我们如何将依赖于第三方SDK的业务逻辑（Java）从任务中物理地剥离出来。我们曾经探讨彻底消除对第三方SDK依赖的可能性，后来发现其成本太高。组件级重用与应用级重用是在建立对问题定义的过程中很重要的两个概念，典型的应用范例是微软将二进制组建模型从COM升级为COM+。SOA是另一个发挥重要影响的概念，我们将业务逻辑封装为本地服务代理，并以标准的方式暴露其接口（WebService）。

结束语

以系统的方法来理解创新思维的基本方面有助于我们了解持续创新的内在规律。上述分析结合了我近几年的工作体验与思考，并试图展现出一个循序渐进的创新思考模型。创新思维之复杂原本不是我所能涉及一二的，但最终还是决定把目前的认识提出来，作为大家进一步讨论的起点。

许正华

目前就职于CA Technologies中国技术中心，从事数据备份和高可用企业软件方面的开发和研究，并专注于软件工程、知识工程、团队文化等领域的研究与实践。

责任编辑：杨爽（yangshuang@csdn.net）

关于本文的论战

作者意在讨论方法论本身，而非某一具体的方法。从这一点来看，作者的逻辑是：创新需要批判，批判需要问题，问题需要概念模型；概念是从对认识的持续批判中得出、修正与进化的。那么在这一逻辑中，所谓“创新”不过是一个具体方法的结果，甚至是对具体结果的修饰而已。作者对方法论的讨论未能最终落足于“如何‘认识’一个系统”，而是趋向于迎合读者口味地去强调“创新”，这是本文的一大弊处。但文章总体的可读性、系统性仍是可供读者品评的。

——周爱民

对于周爱民老师给出的点评意见我基本上同意，除了关于迎合读者口味地去强调“创新”的评价，至少它不是我的本意。我想讨论的是创新思考的方法本身，而不是创新的结果。现实中人们更关注于结果，而忽视了创新性思考其实只是若干可以被训练的思维方法中的一种。创新本身并不神奇——就像人们在谈论所有天才那样，如果我们也能够看到他们艰辛的成长道路与所使用的先进训练方法，就不难理解“其实天才也是可以培养的”。

——许正华

Marty Cagan

过去20年，Marty Cagan作为负责定义和开发产品的高级经理人为多家一流企业工作过，他亲历了个人电脑、互联网、电子商务的起落沉浮，致力于通过写作、演讲、培训帮助客户打造富有创意的产品。

如何获取用户需求

文 / Marty Cagan　译 / 唐丰能，林云源

Marty Cagan是享有世界声誉的产品管理专家，曾经担任网景副总裁、eBay产品管理及设计高级副总裁。本文是他回顾自己二十多年来从事软件产品管理工作的总结和经验分享，分析了现有市场调研方法的利弊，并指出特约用户的重要性。

市场调研

市场部门与产品部门往往存在意见分歧。争议焦点是市场调研工具和市场调研方法在探索（定义）产品中起的作用。这些工具和方法是用户研讨会、用户调查、产品使用分析、拜访用户、可用性/现场测试、同类产品分析。

产生分歧的原因是双方不清楚市场调研的作用与局限性。市场部门夸大了市场调研的作用，而产品部门只看到其局限性。这就造成两种结果：有的团队忽视市场调研，产品偏离了正确的方向；而有的团队又太依赖市场调研，陷入了偏执。这个话题涉及的内容很多，这里我只讨论市场调研方法的作用和局限性。

过去十年，市场调研方法取得了长足进步。随着技术发展，大规模调查用户和顾客的难度降低，很多过去无法解决的问题现在都已经解决了。新技术甚至可以帮你分析用户活动和行为——他们是谁，他们用产品做什么。尽管如此，市场调研最基本的、固有的局限仍然存在，了解这些局限很重要。

市场调研的作用

先介绍一下常用的市场调研工具和方法。

用户调查：网络降低了用户调查难度，提高了调查效率，所以现在几乎要求所有产品都做用户调查。有两点需要注意。第一，设计调查问卷不是一件容易的事，需要技巧和经验。要结合具体情景，认真设置问题，如果调查问卷措辞不清、先入为主，其他部门的同事就会质疑调查结果。第二，调查结果为获得解决方案提供了一条途径，但不是解决方案本身。哪怕所有用户都回答喜欢X特性，我们还是可以通过提供Y特性更实际地解决他们的需求。

产品使用分析：如果你的产品是网站，那么有很多实用的工具可以帮你分析用户访问网站的行为，正确安装和配置后即可使用，非常划算。应尽早使用分析工具，不断地观察分析，然后调整产品。如果你的产品不是网站，那么可以在产品中添加分析工具，记录用户使用产品的行为。应该明确告知用户分析工具的用途，声明只收集统计数据，不涉及用户隐私。这样做虽然麻烦，但很值得。

数据挖掘：收集数据的渠道很多，除了上面提到的产品使用分析，还有用户的账单和账户信息、产品数据等。新的数据分析工具的功能越来越强。如果想知道同时使用几项服务的用户性别比例，或特定人物角色的活跃程度和分布情况，那么通过新的数据分析工具可以轻松获得。

拜访用户：前往用户使用产品的场所（家、办公室）实地考察的作用是任何方法都无法替

代的。虽然拜访用户成本高、耗时长，但每次都能收集到从其他途径无法了解的信息。不过，出于对资金和时间成本的考虑，建议谨慎使用。

人物角色：我喜欢在定义和设计产品的过程中使用人物角色。市场调研也可以借助人物角色展开。请记住，你要面对的绝不是单一类型的用户，务必找出若干主要用户类型，深入了解他们，弄清哪些是当前用户，哪些是潜在用户。

可用性测试：我主张尽早、反复地进行可用性测试。观察用户使用现有产品的反应，收集反馈意见，了解他们的真实想法。从用户的视角重新审视产品，不仅要阅读反馈信息，更要观察、记录用户的行为和反应（比如兴奋、沮丧）。有些工具有远程功能，可以在异地进行可用性测试，记录和分析用户行为。

同类产品分析：产品团队常常低估竞争对手。而实际上，每款产品都有做得好的地方，所以有必要找出竞争对手的优势，学习对手的成功经验。

合理利用市场调研工具和方法可回答几个关键问题：谁是目标用户？用户怎样使用产品？用户能想明白怎样使用产品吗？障碍在哪里？用户为什么选用你的产品？用户喜欢产品的哪些特点？用户希望如何改进产品，增加哪些功能？

市场调研的局限性

最根本的产品问题是：打造什么产品？市场调研结果可作为研发产品的依据和参考，但不能决定产品研发方向。

探索（定义）产品的过程要回答的问题是：采用什么技术来更好解决产品要解决的问题？设计什么样的用户体验？

虽然市场调研很重要，但我从来没听说过仅通过市场调研就成功定义出来的产品。Google、eBay、iPod、iPhone、Facebook、MySpace都不是仅通过调研产生的。

成功的产品基于以下两点认识：深入理解用户需求，以及明白什么样的解决方案在现阶段可行。

我也希望可以简单地根据客户的要求设计产品，但这样做会陷入不断添加新功能、修改产品架构的怪圈。你会为缝缝补补的任务疲于奔命，无法思考创新的解决方案。

如果你的产品已经面市，有一群活跃的用户，可以多与他们沟通，了解他们喜欢产品的哪些方面、不喜欢哪些方面，据此逐步完善产品。但这些建议只能用来修补现有产品的不足，不能用来定义新产品。

总而言之，市场调研结果可以用于完善现有产品，精益求精，但不要指望从市场调研中发现下一个FaceBook、Flickr或者YouTube。

特约用户

大部分市场人员认为拥有一群忠实的、乐于推荐产品的用户会让产品发布变得更容易、效果更好。如果是平台产品（为他人在平台上开发和部署应用提供支持），最好还提供一批起示范作用的应用程序。但让我吃惊的是，许多平台产品在这方面几乎毫无准备。

如果有若干知名人物公开声明使用过产品，并且表示满意，就可以大大降低潜在用户的顾虑，营销推广工作也会容易很多。反之，如果缺少用户推荐，再高明、再新颖的销售策略所起的作用都有限。大众一般对无人问津的产品非常谨慎，要么怀疑其质量，要么认为还不成熟，谁都不愿意第一个吃螃蟹。

如果面向大众的产品只有一两家网站推荐，那么很容易让人误以为这是针对特殊用户群的定制产品。无论是平台产品、商业应用，还是针对大众的互联网服务，用户都希望看到他人使用的效果后再尝试。

接下来让我们把关注的重点从产品发布阶段转移到项目最开始的阶段上。

产品经理要深入了解目标用户，明确产品需要解决的问题，定义满足用户需求的产品，因此产品经理必须和用户紧密合作，这样，开发的产品才可能满足更广泛的用户需求。但问题在于，同时和这么多用户打交道显然是不现实的。

为了解决这两个问题——既深入洞察目标用户的需求，又赢得用户对产品的推荐，我建议征集特约用户（也称用户顾问委员会、用户评审团）协

助完成产品研发。这不是什么新方法，我二十年前就在惠普用过。

方法很简单，在项目的开始阶段物色至少6位积极、活跃、乐于分享的目标用户（可以先招募8-10人，然后从中筛选），要求是他们在产品的目标用户中具有一定影响力。至于他们是否使用过公司原有的产品并不重要，只要他们认为未来的产品可以解决他们手头亟待解决的问题就行。

成为特约用户的好处：参与构思产品创意，解决正在困扰自己的问题；提前使用产品，尽早解决问题；提前使用产品，显著降低各种成本。

产品经理的收获

■ 聚拢一批积极的用户，因为他们可以为定义产品和开发产品提供建议和协助。

■ 提供调研便利，便于产品经理去特约用户的工作场所调研。如果是平台产品的话，便于产品经理去开发人员的工作地点调研。

■ 可以定期组织特约用户进行小组讨论。

■ 特约用户第一时间试用、测试产品，迅速反馈意见。

■ 如果特约用户满意产品的表现，会乐意公开推荐产品。

组织特约用户的注意事项

■ 不要收取参与费用，否则合作关系会变味。产品经理需要的是开发产品的伙伴，不要变成为特约用户开发产品。如果特约用户愿意，你可以等正式产品发布后再向他们收取费用。

■ 由于可以免费试用产品，通常会有大量的申请者申请成为特约用户。公司的销售部门为了提高业绩，可能会要求产品经理招募更多的用户。这会消耗产品经理大量的精力，而且这些用户不一定符合要求。为了满足大批心急的用户，公司可以发布预览版产品。特约用户的人数绝不能超过10个，否则产品经理不可能有时间和精力与每位用户深入沟通。

■ 如果在寻找特约用户时遇到困难，很可能是因为产品要解决的问题不像产品经理想象的那么重要，将来也很难销售出去。这可以初步验证产品创意是否有价值。出现这种情况，产品经理应该重新考虑产品计划。

■ 产品经理需要确保特约用户是产品的潜在目标用户。我们很容易把产品尝鲜者（early adopter）误当成特约用户。产品尝鲜者常常能容忍产品的不足和缺陷，根据他们的建议研发的产品，很可能只适合他们自己，无法满足大众的需求。

■ 产品经理务必向特约用户说明，要开发的是面向大众的通用产品，不是为某家公司开发的定制产品。特约用户也不希望出现这种情况，因为小众产品的生命周期比较短，一旦产品被淘汰，售后服务也将被取消。产品经理应该向特约用户承诺产品不会昙花一现。

■ 产品经理应该把特约用户当成开发伙伴对待，视他们为同事，互相帮助。许多特约用户和我结下了深厚持久的友谊。

■ 产品经理与特约用户的合作贯穿产品研发的每个环节：向他们展示产品原型，请他们参加测试，向他们请教产品的细节问题，让他们帮你部署、测试待发布产品的备选版本。

■ 正式产品发布之前，一定要先请特约用户试用，确保每个人都满意，一旦发布，他们会坚定不移地向大众推荐产品。

■ 产品经理要和产品营销团队紧密合作。一方面，营销团队可帮忙物色特约用户；另一方面，可协助产品经理提高特约用户受关注的程度。

■ 如果是平台产品，特约用户的作用就更突出了，只不过6个特约用户要换成6个应用。产品经理要与特约应用的开发者紧密合作，确保在平台上构建的应用让用户感到满意，最好鼓励应用开发者发展自己的特约用户。

该不该与用户交流

经常有产品经理抱怨，公司不允许他与用户接触。导致这种情况出现的原因很多，可能公司认为应该由营销人员与客户打交道，也可能公司担心产品经理言论不当或者向用户做出不恰当的承

诺，还可能销售代表怕人抢了他的客户，或者产品是通过特殊渠道销售的。不管什么原因，如果公司不允许你直接与用户交流，你一定要尝试改变这个规定。如果不成功，我建议你翻出自己的简历另谋高就，寻找可以大展身手的地方。

我无法想象不深入理解用户需求，特别是在禁止与用户面对面交流的情况下，产品经理怎样打造出让用户满意的产品。

产品经理亲自与用户交流的作用是任何市场调查方法都无法替代的。我认为产品经理应该尽量亲自拜访用户，与用户交流，参加每一次的可用性测试和特约用户讨论会，与用户充分沟通，挖掘每个人的潜在需求，捕捉产品创意。“重新定制这个页面，同时记录各类用户的访问量”，“72%的用户要求提高视频的分辨率”，从这类呆板的报告里是不可能洞察用户需求的。

产品经理还应该充分利用公司的可用资源。许多公司设有用户研究部门，可以协助分析用户行为；营销人员可以协助确定产品定位和宣传计划。我个人还喜欢邀请主程序员参与前期定义产品的工作，让他提前考虑产品开发的细节问题。这些工作都可以请人协助，唯有理解用户需求的工作，产品经理不能推诿他人。

最后补充一点，与用户打交道的过程中，你会发现一些富有洞察力、善于思考的用户，应设法与他们建立长期联系（记下他们的联系方式）。他们是特约用户的最佳候选人。

注意，虽然我提到的例子多数是企业软件和平台产品，但这些方法同样适用于针对大众的互联网服务和消费类产品。如果是互联网服务，特约用户的人数应该增加至10-15人，不过要保证产品经理有精力充分了解每位特约用户，以及他们使用服务的环境（家或办公室）。设计和规划网站时很容易犯一类错误，即对用户需求把握不够，收集的用户反馈信息不充分，直到项目尾声才发现产品的定位有问题。这样做风险很大，而特约用户可以帮助产品经理把握用户需求。

另外，从营销的角度看，大众消费者对口碑营销的反应可能与企业采购经理不同。大众消费者更容易受媒体和网站评论的影响，但他们也希望看到真实用户的评价。在这一点上两者是相同的。

特约用户是确保产品不偏离用户需求最简单有效的关键，同时也能向潜在用户宣传和推荐产品。

关于用户研讨会

上文没有介绍用户研讨会的作用和局限性，因为我不能肯定其功效。我支持产品经理多接触目标用户，组织用户研讨会可以让大家面对面交流，但只有谨慎筹划，才能发挥其作用。

虽然组织用户研讨会可以面对面了解目标用户对产品的看法，但我保证在用户研讨会上不可能讨论出成功的产品，有以下两个根本原因。

- 用户不知道什么样的想法是可行的，多数用户对现有技术一无所知。
- 用户不知道自己想要什么，没见到实际产品，用户很难凭空想象自己需要什么。

用户研讨会还存在以下一些弊端。

- 人群聚集时容易冲动，相互影响，难以获取每个用户的真实想法，取而代之的是那些善于表达者的一家之言。
- 只有让用户试用产品，他们才清楚自己想要什么，而通常组织用户研讨会时产品还没有眉目。
- 与用户调查一样，组织用户研讨会也需要经验。主持人不但要熟悉组织技巧，能随机应变，还得掌握产品领域的知识，擅长引导话题，才能获得预期的效果。否则会收效甚微。

如果必须组织用户研讨会，务必让产品经理参加。布置会场和后勤服务的工作可以外包，但收集信息和分析数据的工作绝不能外包。

本文节选自华中科技大学出版社《启示录：打造用户喜爱的产品》一书及作者的博客。该书从人员、流程、产品三个角度介绍了现代软件（互联网）产品管理的实践经验和理念。特此感谢华中科技大学出版社与Marty Cagan先生授权。

怎样营造良好的技术文化

众所周知，良好的技术文化不仅有助于人才成长和提升团队内聚力，更是吸引和留住优秀人才的一大法宝。然而，该如何营造良好的技术文化呢？且听本期三位嘉宾的经验分享。

张克军
豆瓣前端团队负责人

悉心营造良好的技术文化

工程师的荣誉感

存在主义认为，文化是对一群人存在方式的描述。一个技术团队能长期存在下去，一定会产生某种技术文化。也就是说，技术文化会自然产生，受人的因素影响而变化。随着团队规模的变化，团队的技术文化可能变好或变坏，可以通过以下一些表象来衡量。

- 技术上的活跃度。
- 高质人才的密度。
- 工程师的满意度和荣誉感。

良好的技术文化能给工程师某种荣誉感，自然满意度也不会低。这种荣誉感里带有某种优越感，比如会觉得其他公司的技术很“土”，理念落后、工具落后等。一种技术文化中会包含某种共同的规范、意识、价值观和做事原则等，新加入的人如果认同，便会因受其影响而很好地融入到团队中。

“净化空气”很重要

技术文化好比空气，营造良好的技术文化就是改善空气质量。降低“污染物”排放是第一要务。而哪些是“污染物”呢？

- **过重的管理行为**。管理的目的是保证产品开发按计划有序进行，而过重的管理行为会适得其反。
- **不当的开发流程**。这是一个槽点，草率进入开发环节，会致使开发进程不是一种有序的迭代，经常由于随意变更需求而做无用功。
- **急功近利的心态**。为了攒业绩，不管质量、不愿意同他人协作，或单纯追求本部门业绩，不为合作方考虑。
- **不适合团队的人**。不适合团队的人不一定是能力不行，主要是价值观跟团队技术文化中的价值观相悖。这些人的负面作用远大于他们的贡献，应该果断淘汰掉。

提升团队内聚力

对于快速发展中的团队，注重团队技术文化塑造是团队进一步发展的基础。它的基因往往是最初的几个人或十几个人决定的。这方面跟企业文化的形成没什么不同。但如果在初期没有将其中的精华提炼出来，那么随着团队新人增多，技术文化会很快被稀释，而且人越多越难向好的方面导向。在Facebook创业初期，创始人Mark确定了几条很精练的工程师的行为准则，并贴在墙上。这些准则体现出了技术上务实、践行的风格，它吸引了一些同样

有如此风格的优秀工程师。整个技术团队的内聚自然越来越强。

对于快速发展中的团队，注重团队技术文化塑造是团队进一步发展的基础。营造良好的技术文化，需要在以下几方面努力。

■ **树立行为准则**。它虽然没有强制性，但在各环节之间协作、各角色之间协作中出现分歧时，它是很好的参照物。因此它要有这样的效力，而不只是写在纸上的标语。

■ **注重技术积累**。技术积累是技术文化的底蕴。认真对待每一次项目实践，不以解决问题为目的，而是在解决问题的同时，将经验记录和积累下来。

■ **奖励贡献，包容错误**。激励制度是不可或缺的。工程师对团队的贡献要有荣誉和物质上的双重奖励，激励才能有效。包容错误，才不会让工程师顾虑重重，因为只有突破陈规、大胆尝试，才能有所收获。

■ **人人都要有工程技术思维**。良好的技术文化的形成不只是技术团队内部的事情，它也受企业风气影响。不合理的开发流程、不适当的行政制度等，都会让所有试图改善技术文化的努力付之东流。

营造良好的技术文化，就好比孩子性格的养成和心智的发展，它将决定这个孩子未来的命运。

许晓斌
技术作者，敏捷咨询师，持续集成专家

分享促进技术文化建设

促进分享是建设技术文化非常重要的一环。程序员一方面希望展示自己的所学并获得大家的认同和尊重，另一方面也希望学习他人的知识以提高自己的技术能力。我在Sonatype工作时，技术出身的老板经常写博客分享技术，同时也鼓励员工每周花2～3个小时写技术心得，或者是直接在博客上写零散的产品用户文档。这样，不仅同事之间多了一个相互学习的渠道，也让外界了解到了这个公司的技术文化。

促进分享的方式很多，下面仅列举了几种。

■ 提倡团队成员写wiki和博客，并鼓励大家相互帮助。比如，让写作水平高的人帮忙编辑他人的文章并对外发布，还可以进一步投稿到专业技术媒体。

■ 鼓励团队成员读书，并定期组织一些简单的内部研讨会。注意，不要占用员工的休息时间，那样会适得其反。

■ 邀请领域专家到公司内部做分享。在积累技术知识的同时，还能收获很多新的观点。

■ 鼓励团队成员参加技术会议。如果有人能到技术会议上做演讲和分享，那就更好了。

■ 参与开源项目。鼓励团队成员在使用开源产品的同时，回馈开源社区，比如提交补丁。如果有条件，甚至可以考虑开源公司的内部组件或项目，这样不仅能收到更多用户的反馈，也能给团队成员带来极大的成就感。

分享的气氛一旦建立起来，还需要大家一起认真保护。对他人的工作成果我们应当持感激和尊重的态度，认真学习并给予及时的反馈。在同事花了好多天业余时间认真准备，然后花一个小时演示一个新的技术后，一句简单的“好”或者“不好”都是对他的不尊重，正确的做法应该是先表示感谢，然后列出欣赏的地方，同时指出认为可以改进的地方，当然，若有疑问应当进一步讨论。

良好的技术文化不仅能让大家工作得更愉快，而且能促进沟通，帮助打破一些日久生成的樊篱，使日常的工作变得更方便和轻松。

庄卓然（南天）
天猫（淘宝商城）产品技术部资深经理

技术文化建设实践

良好的技术文化就像一个稳定和平衡的生态环境，有助于工程师以有效的方式思考和学习技术，促进自身的成长。同时，工程师技术上的每一点改进和创新都能够为客户带来更好的用户体验，也能为公司带来更大的收益。然而对于技术团队的管理者而言，要想创造好的工作环境，并不是一夜间就能完成的。它需要核心架构师和管理者的长期投入。不同的公司和团队在不同阶段对技术文化的要求不同，文化本身很难界定好坏，只有适合与否。本文仅介绍一些好的实践。

学习方向和工作相结合

很多团队常犯的一个错误是，团队的学习和分享没有明确的目标，只讨论眼下流行的技术，却忽略了团队的方向和正在做的工作。这种看似百花齐放的学习氛围，其实对大多数团队来说是一种浪费和伤害。

首先，由于工作中用不到所学的技术，所以很难有时间和场景深入地研究下去，致使大部分学习只能浅尝辄止。其次，新技术层出不穷，但绝大多数是新瓶装旧酒，本质上的思想并无创新。如果错误地认为很重要并且迫切想要学习，那么势必会事倍功半且不能做出有用的东西。再次，以这种学习方式衍生出的分享很难创造价值。分享者学习得不够深入，很难分享出有价值的内容。而其他人也只是看个热闹。久而久之，失去了分享的意义。

我比较喜欢的形式是圈定几个具体的领域，通过技术兴趣小组的方式自由组合，在一个实用的方向上深入学习。

沙龙和分享是常见的形式，也可以通过一些较低风险的项目试验想法和新技术，再配合wiki和定期的邮件作为沉淀的载体。绝对不要因为模糊不清的兴趣去组织活动和学习，时间和精力很宝贵，应当谨慎分配。

让参与和分享成为习惯

对技术积累有价值的事情，尽量全员参与，而不提倡一个人完成绝大多数工作，其他人只是被动地接受信息。

无论是代码审查还是阶段性的项目总结，都要让项目团队的所有工程师分享观点，共同去分析和讨论。小到一个经典Bug的分析和修正过程，大到某个产品的技术创新，都要尽量鼓励团队成员公开并分享他们的成果。不用拘泥于形式，公司可以提供分享平台，员工也可以自发进行组织，一封邮件、内网的一个帖子均可。这会慢慢产生效果，形成积极分享的氛围。对于代码细节更应如此。

这种分享和参与同样可以延续到公司，比如参与开源社区的技术讨论和具体产品，在各种技术大会上分享团队经验，让更多的团队成员意识到，我们不仅可以依靠公司的平台改变用户体验，还可以通过影响和我们一样的工程师改变更多用户的生活。千万不要低估这种意识的力量，它会激励每名团队成员不断地追求卓越。

润物细无声

良好的技术文化不是凭空出现的，它是自然生成的，是团队一贯行为的副产物。如果鼓励成员分享，那么分享就会成为团队技术文化的一部分；如果不断追求更健壮的架构和更优秀的代码，那么这也成为团队技术文化的一部分；如果鼓励团队之间的紧密合作，那么合作就成为团队技术文化的一部分。

技术文化是团队经过磨合和历练后形成的共同需要。因此，不要过多担心，更不要去强求，从最重要的小事作起，让良好的技术文化像佳酿一样慢慢发酵。

我们所做的一切都是为了创新

青蛙设计首席创意执行官Mark Rolston专访

特约撰稿人 / 西乔

青蛙设计首席创意执行官Mark Rolston在接受《程序员》特约撰稿人西乔的采访时谈到，软件是设计的首要推动力，中国市场也正在意识到需要高品质的设计来帮助产品脱颖而出，并预测数据可视化将成为设计的重要分支，而数字世界和自然世界最终将合二为一。

在创立之初，青蛙设计的目标就是把设计定位为策略性专业与工业和商业相结合，创造出审美和功能兼备的科技产品。1982年，青蛙设计获得了和苹果合作的机会。它提供的设计背离了当时科技产品笨重、单调的外观，提供了一种新的设计语言，其中包括如下一些策略：苹果电脑将会是小巧、干净、白色；所有图形和字体都必须是简洁而有秩序的；最终的外观将由最先进的工厂车间打造，具有灵巧、高科技感的特点。30年过去了，我们可以惊讶地发现，这些策略仍然作为苹果风格的灵魂，沿用至今。今天的青蛙设计早已不再称自己为一家设计公司，而是一家创新公司。正是对创新的不懈追求，加上远见和冒险精神，让青蛙设计从一个间工作室，成长为今天全球尊敬的国际设计巨头。

作为首席创意执行官，Mark Rolston（以下简称Mark）负责青蛙设计的全球创意构想，是数字媒体、用户界面、设计、电子商务和移动应用领域国际公认的专家。作为一个互联网先锋，他于1996年创建青蛙设计的数字媒体部门（Digital Media Design group）。在2000年推动了Dell官网的创新，让其成为迄今为止赢利最高的网站之一，树立了电子商务新标准。又与i2、微软、SAP及SUN等品牌合作，不断改进人们的数字体验。

Mark Rolston在1996年创立了青蛙设计的数字媒体部门

软件是设计的首要推动力

西乔：不少中国人第一次是从《乔布斯传》中知道青蛙设计。20世纪80年代和乔布斯的两次合作的设计，获得《时代》周刊评选的年度最佳设计奖。与苹果及乔布斯合作的经历给青蛙设计带来了什么？

Mark：1982年，Esslinger（青蛙设计创始人）开始为苹果工作，这段合作经历让我们有机会参与设

计了里程碑式的产品。令人印象最深刻是，这种合作关系的性质不是两家公司之间的商业关系，而是像真正的搭档和朋友那样合作。

乔布斯能够敏锐地发现别人的价值，尊重你的价值，和你成为伙伴。人们都知道设计师可以提供重要的价值，但往往只有那些拥有开放价值观的客户才能从中受益。

我们一直都在帮助客户创造有影响力的产品、服务和体验，给他们带来比预期更多的东西，让他们有更多兴趣去挑战。苹果公司是从这类合作关系中受益的一个案例。

西乔：您在1996年创立了青蛙设计的数字媒体部门，这在当年是很有远见的。是什么样的想法促成了您来做这件事？

Mark：我一直接触计算机，很小就开始学习编程，同时也是一名设计师。我一直在做软件开发，而且希望利用设计让软件变得更好。任何领域的设计师，都梦想能去改变世界，而在那个年代，改变世界的最好方式是通过设计软件。即使你并不亲自使用这些软件，它们也改变了你身边的世界。而当1994年我来到青蛙设计时，我发现这里拥有我想要的一切条件。

那时，青蛙设计已在设计实体产品方面取得很大成功，拥有近30年的丰富经验。但软件设计做得并不多。最初这个部门是帮助已有的工业设计和产品设计部门去解决和软件相关的问题，但很快我们发现很多问题的核心并不是产品的工业设计，而是产品上的软件。所以事情很快发生了改变。到1998年，青蛙设计把注意力转移到了软件设计上。

当时，还有许多人质疑为什么我们投入如此多的精力在软件上。但到2000年时，人人都知道软件是未来，将会是设计的首要推动力。

数字世界和自然世界将合二为一

西乔：当年的数字媒体设计部门现在已发展为青蛙设计的主营业务之一，这充分证明了您当年的前瞻性。作为一个数字设计领域的先锋，在16年后，您对数字化未来有什么预见性看法？是直觉式（intuitive）的设计吗？

Mark：计算机已给人类和社会带来了本质改变，但目前人们还是必须借助特定的设备来操作计算机。你必须拥有一部智能手机，或者桌子上必须放着一台电脑，这些设备是我们通往软件世界的走廊。但同时它们和自然世界也是分离的。你需要特殊的技能，在特别的场景下去操作它，你离不开按键、屏幕、输入、输出。我认为计算机应该和我们的使用情景融为一体，而不是一个需要转换焦点去进行特殊操作或交互的对象，所以我认为这些设备在未来可能消失。在未来，当我需要打电话或搜索信息时，我能够直接利用身边的环境，用自然的方式来完成这些任务。

不仅是更直觉式的操作方式，而且是真正把数字世界和自然世界合二为一。计算机会变得更加适合与人类交流。交互方式会变得更自然，更符合人类习惯。而触屏技术正是实现这一趋势的突破。在未来，计算机会拥有自己的意识，能够不断学习和识别周围的一切，我们可以和它共处一室，使用语音指令，像与人对话一样在3D空间里和计算机进行沟通。

西乔：您认为未来的终端不会有固定的外形，它将可能存在于任何情境中，是任何东西对吗？

Mark：对，这就好比剑。对于一个统治者，剑最初是武器，但后来就变成了力量的象征和对领土统治的申明。剑的实际功能消失了，象征性的意义保留下来。所以现在我们所理解的终端也会这样，形态会消失，但延伸价值会保留下来。今天我们拥有这些设备，就像剑一样，给你力量。人们也将不再只通过具体的功能来标识某一物理事物。它可以成为任何你希望它成为的东西。人机交互也不再被绑定在某一设备上，而是和环境完全地融入在一起。这一切都是由人机交互界面的快速发展所带来的。我认为人机交互会经过5个阶段的发展。

- 计算机成为人们生活的一部分。
- 我们随身携带它。
- 我们通过它来感知世界。
- 计算机装备在我们身体上。
- 我们就是计算机本身。

如今，我们正在给物理世界打上各种标注；我们身边到处都能看到各种形式的计算机、个人电子设备；我们的身体正在变成节点、外围甚至是交互界面本身。也许当我们正在谈论的未来在某一天最终到达时，它可能看起来很理所当然。但时尚不正是把今天看起来奇怪的一切变成明天被普遍接受的事物吗？

当一个设备可以记录你的健康指数，同时也用于更新你的SNS状态时，想象一下，会发生什么？你的心跳会成为你与别人对话的一部分。

西乔：它们最终会成为人体的一部分吗，比如植入人体？

Mark：当然。但需要花上较长的时间。有可能在未来十年内，开始有小规模的实验，例如植入式的无线电通信设备。假设一下这个场景：当我们想非正式地获得关于我们身体的信息，可以往手臂里植入很小的医疗设备，然后可以得到一些预测信息：例如前一夜的醉酒会让我今天的身体发生些什么？

西乔：您在2009年关于《The White Box》的演讲中，讨论了移动相关的技术和设计。在那次演讲里您提到了“第六感设备”（The SixthSense Device），这个概念是在Pranav Mistry的TED演讲中所展示的一种人际交互设备。您记得它吗？

Mark Rolston认为计算机最终将成为人类身体的一部分

Mark：是的。第六感是一个早期的实验性设计，拥有很好的创意。我们与他们有合作。

西乔：但有人质疑它的可商用性和使用精度。

Mark：我们一直在推进它，寻求更明朗、更高水准的解决方案。 我们尝试将交互界面投影到更多不同的环境中，和新的应用场景结合。而且现在投影设备也有了更好的选择，每一年它都在变得更好。你应该对这个世界充满想象力。这类设备的识别精度，准确度和运算能力都在提高。看到的东西越多，识别能力越强。就像Google，用户的每次搜索，都会让系统变得更加智能，让下一次搜索变得更快、更准确。每天由用户带来的上万次使用，会积累成惊人的结果。这就是商用化的前景。现在我们已知道它可以和电子游戏设备进行连接，你可以期待这种连接很快放到笔记本、智能手机和许多设备上。

数据可视化将成为设计的重要分支

西乔：最近有很多国内外的设计Blog都在推荐青蛙设计最新的一个案例：Bloomberg's Big Beautiful Data。它使用了HTML5和实时数据，在风格上也很容易联想到Metro UI。请问您怎样看待Metro UI在设计和交互方式的突破和适用性？

Mark：我认为微软是一家高品质的公司，在某些独特的领域中取得了非凡的成绩。但随着全球层面的产品革新风潮，如果只维持传统风格，它们将会受到很大挑战。微软过去太执着于修补已有的产品，缺乏突破性，如果希望赢得人们对它们设计的尊敬，类似现在苹果公司得到的，还是要走出自我的限制才行。 如今微软能够走出这一步，创造一种属于自己的新的设计语言，是非常好的事情。

Metro UI这套设计语言，要求这个平台上的应用都要符合这种风格，这相当于限制了边界。开发者需要接受良好的训练才能掌控这种风格。成熟的框架是Android和iOS平台成功的原因之一：通过设定一套设计语言，任何文化下的开发者都可

以避免混乱，开发出和平台相协调的应用。但同时也有很多弊端，比如增加开发者的学习成本。当希望创造一些框架中没有的独特东西时，也要付出很高的代价。

西乔：这个设计案例同样采用了数据可视化（Data Visualization）的表现手法。关于数据可视化的应用前景，您怎么看？

Mark：数据可视化会变得非常重要。在过去，数据的获取是有限的，人们也只能利用现有的数据来产生和获取价值，但现在我们拥有了历史积累的海量数据，去解释和分析它们具有重要的意义。世界上大部分数据是未经加工的，我们要做的不仅是把它们图形化，而且是通过结构化数据，对他们进行组合、运算，提炼出更深的含义与价值；不仅是了解个体数据，同时也要获得对整体模式的认知。伴随着数据存储、处理与分析能力的提高，未来它会是一个很大的产业，成为未来设计的一个重要分支。对于所有人，这都是新的挑战和机会。

所有市场都期待高品质的设计

西乔：青蛙设计在官方网站上提到有一个专长是“中国”，这说明你们对在中国市场的业务非常有信心，对吗？

Mark：我们对任何地区的市场都很有信心。中国是一个每天都在增长并且快速改变的市场。我们在这里花费了大量时间，围绕中国用户的行为和消费者的生活做研究。我们关心他们的想法，他们如何生活，他们的需求是什么，他们希望世界变成什么样？在零售、医疗、消费电子、移动通信、媒体、时尚等诸多领域，我们积累了大量对用户的了解。中国是一个很大的市场，拥有大量的用户和快速变化的节奏。对于该如何开展在中国的业务，引导中国的用户，我们还处于学习的过程中。

西乔：有人总结，在中国，设计的最大的特点是山寨和没品。从历史上看，中国制造的品牌并不重视设计的价值，因为它们的优势就是低研发周期、低成本、低价格竞争，并不在乎拿用户冒险。在这一点上，青蛙设计对待设计的观点，或者说青蛙设计提供的服务，和这些“中国制造”的成功之道并不相符，甚至与它们的节奏和优势是冲突的。在这种情况下，你们仍对中国市场充满信心吗？

Mark：我们相信，任何时候，任何市场，人们都想要高质量、有意义的产品。用户和市场期待突破性的改变。当桌上有20个看起来差不多的手机时，用户的选择会很困难。但如果其中的一款手机做了更好的设计、使用了更好的材料、配备更好的软件，用户的决定将是显而易见的。每一个市场都证明了这一点，如果人们有足够的资源，他们自然会选择更好的，去赋予生活更高的价值。在中国，经济发展使人们拥有了更高的消费能力，去选择拥有出色设计的产品。人们会为了追求更高的品质而花钱。这是伴随经济增长自然而然发生的现象。

我们也接触过制作山寨产品的客户，他们不再想只做山寨产品，而是想要好的原创设计。他们意识到这会给他们带来更大的市场。

另外，中国通常认为他们所面临的版权现状是很特殊的，但所有年轻的市场在开始时都伴随过这种问题。在19世纪，美国也盗版英国的图书而不支付任何版权费用。在不同的历史时期可能表现有所不同，但市场和用户成长的趋势是相同的。品质、创新、标准和清晰是人类共同的追求。

同时，山寨工厂也不仅仅只做了抄袭，它们也从这一过程中获得了技术和经验。最后，技术也会被用于去创造新的东西。市场会激发企业，激发用户。人类与生俱来就拥有创造的激情，这种激情必定也会在中国成长起来。

西乔：中国市场还有一个特点是，用户群的消费能力、消费习惯和使用习惯差异很大，不同地区之间文化差异也很大。你们网站上提到，你们能够分析中国一线城市和农村市场中消费者的行为，来作为创新机会的灵感。在对待中国在用户群和地区的巨大差异问题上，你们有什么心得？

Mark：虽然这些差异会限制产品的适应性，但我们试着不去考虑这些，而是把注意力放到用户真正的需求上。他们只是不同的用户，不管生活在

大城市还是农村，人们都期待高质量的生活。差别只在于是否有足够的经济能力去得到它而已。

创新是成功的一条捷径

西乔：设计研究和用户研究是青蛙设计的另一个专长，那对设计的研究主要是用于解决客户在发展中遇到的现有问题，还是用于激发灵感、形成创新的产品策略？

Mark：我们所做的一切都是为了创新。设计研究不会确切地告诉你应该去做什么，而是帮助你建立理解和关注重点。最终设计师会从这些灵感中得到信息，获得原创的解决方案。设计研究无法直接回答问题，而是提供一种帮助你解决问题的工具。

西乔：在用户研究中，有两个很难回避的问题：一是用户其实并不知道它们真正想要的是什么，二是统计和数字解决往往并不能真正解读用户的期望。还有人说苹果从来不做用户调研，因为他们做的是超越时代的产品。您同意这种说法吗？

Mark：不，我不这么认为。用户不会直接告诉你他们要的是什么，但如果你多花点时间，就可以了解到他们真正的需求。苹果会做用户研究，但不是以传统的形式。苹果只设计自己会去使用的产品，在设计和开发过程中，设计师、工程师和员工会反复使用他们所创造的产品，在使用过程中不断测试、分析、研究和提升产品。他们自己就是用户，这种关系会驱动他们不断完善设计。而很多其他公司并不是这样，它们会设计一些自己从来不用的产品，也从来没想过要成为这些产品的用户。

对于我而言，我并不一个中国人，但如果我需要为中国用户设计产品，我会亲身实地体验中国用户的生活，来了解他们真正的需要。很多中国公司追求快速的研发周期，不愿意做深入的用户研究。但了解你的用户是非常有价值的事情，你可以变得离他们越来越近。

西乔：在设计研究中，你们主要采取的是定性研究，这会带来一个问题，研究的成功与否非常需要依赖研究者的水平。这就导致这种研究模式难以复制，服务的规模难以扩大，青蛙设计是怎样解决这个困境的，以保持研究质量的稳定？

Mark：我们通过创建和培养企业文化来解决这个问题。青蛙设计有很好的传统与方法去传授它的设计研究思想和创新文化，让员工具备高水准的技巧和能力。同时我们在做设计研究时，不仅只有分析师和研究者，设计师也会参与整个过程，当他们在观察用户时，同时也在考虑方案。青蛙设计在设计方法的一个重要特点正是将设计和研究相结合。

西乔：产品主要关注的是给用户带去的价值；商业主要关注如何赢利；而创新则更关注如何超越期待，获得更高更长久的利益，这三者有一致也有冲突，你们是如何帮助客户平衡它们的？

Mark：我们的很多客户也问过这种问题，但在成功的产品和设计中，这三者是不冲突的。开发一个好的产品，可以让你获得更大的市场，赢得更高的利润。同时一个好的产品也一定离不开创新。创新实际上提供了一种更新、更容易的方式让你赢得市场。iPhone的成功就是最有说服力的案例。因此，解决冲突的方法就是创造最好的产品、最好的设计。

但对于青蛙设计而言，所面对的最大挑战便是去说服我们的客户。他们觉得已花了很多钱在设计上。但这不仅仅是投入多少钱的问题，而是要投入精力，去重视产品和设计的价值。

西乔：中国的客户更难被说服吗？

Mark：是的。但这只是时间问题。市场在不断成长，企业会感到更多的竞争压力。比如华为，当我们第一次与它们合作时，它们并没有感到太多竞争，它们有自己的特长。但如今，它们开始希望通过更好的产品设计来取得更大的优势。有许多成功的公司，都利用设计策略让自己更具个性，在竞争中脱颖而出。不仅是传统意义上的产品设计，还包括利用创新的方法和程序去改造生产流程。重视产品品质和创新，对于用户和企业是双赢的，既改善了普通人的生活和产品体验，也为世界带来更多价值。

情感牵引和交互在游戏中的价值

文 / 郑金条

本文详尽阐述了以关系链和情感沉浸为基础打造的情感牵引和交互在游戏中的价值，作者认为，捕获用户在情感线层面的投入已成为未来必需的层面。

游戏的未来趋势有三个层面，其一是简单闲趣的偏单人游戏，例如《Temple Run》、《Doodle Jump》，它们将长期把控手机游戏的付费榜单。其二是随着用户对交互的需求和Facebook社交游戏向移动端迁移，手机社交游戏将成为交互类游戏角逐的核心战场，诸如Zynga、Crowdstar或Funzio。其三是手机网游，App Store中国区的营收榜就很有代表性，典型的诸如《神仙道》以及三国主题系列游戏。

在未来的手机游戏层面将重新凸显出两种基于用户黏着度的方式：用户的情感沉浸和基于关系链的社区交互。

玩家离开游戏的相关缘由，除了游戏机制和题材方面的困惑（可玩性不足、趋于重复式或强迫式操作、素材缺乏新鲜感），以及在画面与声效方面缺乏表现力之外，玩家间因为游戏频度差异出现过大的进度差距或者关系链断裂失去社区交互价值也是其中需要权衡的环节。

基于关系链的社区交互带来的影响

Zynga首席执行官Mark Pincus在界定社交游戏时以一个酒会为例重新界定了游戏在交互中所扮演的具体含义。在他的描述中，玩家首先会因与老朋友重会而欢愉，但更让玩家雀跃的是能因游戏偶遇一些值得结交的人，他的终极想法是让玩家和他的朋友以及朋友的朋友在游戏中欢聚。

换句话说，在弱游戏性的环境下，玩家之间彼此架设的黏着效能最终决定着游戏所能呈现的价值，Playdom游戏设计副总裁Steve Meretzky在定义游戏时就认为做好游戏交互氛围以情感纽带才能长久地留住用户，这种想法和Playdom资深设计师Tony VentriceT相似，在他看来，创建一个持久交互的稳定内部社区并驱动和培养用户对游戏未来生命周期的延续性具有决定价值。Shufflebrain工作室的创始人Amy Jo Kim将这层关系说得更为直白：单个的玩家可能烦透某款游戏并从心底否定游戏的可玩性，但因为他的现实关系链好友都在游戏进程中，并且刚好其他玩家的游戏进程需要该玩家予以配合，在这种交互框架下，持续游戏就变成了完成友谊价值的一部分。密歇根大学和巴西佩洛塔斯联邦大学的研究证实了关系链在游戏玩家的相互黏着力方面起着很好的隐性受迫力和牵制力。这种协同关系不一定是好友以固定方式结盟的形式，也可能是密集交互的支撑架构，不管是竞争还是协作，只要他们在游戏中经常以各种形式接触，其社区依赖就可能产生。

我曾基于这个问题探讨过如何营造好友邀请时被邀请者的荣誉感，这也是架构游戏虚拟社区时所需要顾及的一个层面（即被邀请的玩家如何消弭刚开始时的被迫参与感），而在现实生活中我们不会有这样的顾忌，好友之间的互相邀约基本是一件荣耀的事情，无论是家庭聚餐还是简单的小

舞会都能够觉得这是被朋友重视的肯定。这时现实生存状态和游戏间的这种协作出现了严重的脱节。由于好友间的交互让现实纽带已逐步延伸到虚拟网络，就有必要思考如何让游戏中的泛友谊更具互动的针对性。

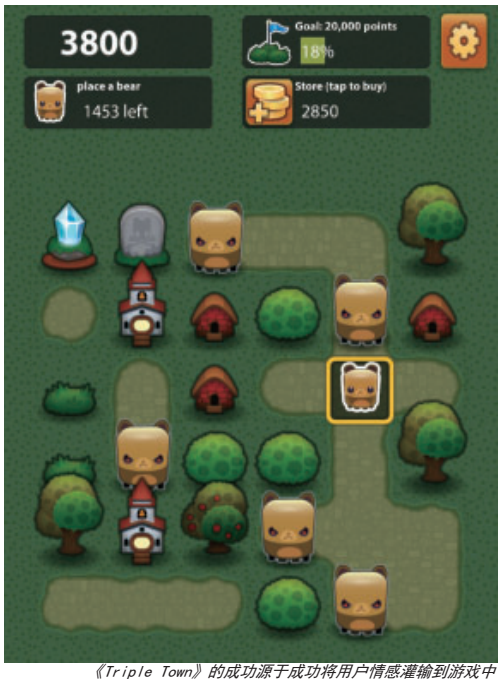
美国罗格斯大学的社会学教授Karen Cerulo认为，尽管这些并不即时且不够严谨的数字化虚拟关系看起来不是那么关键，但仍将因为玩家之间的频繁交互而被明确地定义。就像我们对社交（手机）游戏的私底下认同：一款社交游戏真正的生命力不在于一个用户自娱自乐是否开心，而在于该玩家是否觉得他和更多的朋友一起分享会更有意思。

不管是从Facebook还是从App Store所呈现的交互趋势来看，玩家之间的交互已有从酒会向游乐场（游乐场为Zynga副总裁Bill Mooney在加州大学的表述）转化的趋势，大量“ADD ME”的出现使得游戏交互圈从刚开始的好友和关联纽带逐步突破为更广义的甚至更为直接的纯游戏友情。尽管这一趋势的出现让游戏市场上出现了各种顾忌，诸如社交环境的好友真诚性问题，被冲淡之后的交互圈将失去原先的情感架构和黏性价值，从而稀释社区的归属感。事实上如果单纯从社群感的角度看待问题的话，这种看似虚伪的友情仍然具备极强的拓展游戏交互层面的价值，就像DNB Holdings公司CEO David Barnes所宣扬的一般社交游戏遵循的原则：与陌生人竞争，与好友合作。Ludum Dare创立者Geoff Howland甚至认为竞争也是造成玩家沉浸的一个积极要素。

玩家的情感沉浸

环节1：游戏中的玩家情感维度

International Game Developers Association创始人Ernest Adams认为游戏需要在设定时充分考量玩家在游戏中可能展现的情感环节，或者说玩家是否能够在游戏中找到情感短暂的依托和归属感，在游戏的某些进阶环节，玩家的情绪又将如何变化和引导。Loot Drop Inc创始人Brenda Brathwaite认为好游戏应该让玩家有重回游戏的期待，并在证实了某种期待后产生新的牵引力。



《Triple Town》的成功源于成功将用户情感灌输到游戏中

因此，NGD Studios联合创始人Nicolas Lamanna建议将情感影响力作为评判游戏优劣的指标引入游戏的分级中。一个玩家可能为了升到一个游戏等级或者获得某种特殊的游戏道具而无数次地在刷重复式的任务，在外人看来这是不可思议的，但对于玩家而言，他反倒很享受这种过程。很多情况下，玩家的正常心态并不被认真考虑，事实上情感更能权衡一个玩家对游戏的认可价值。

Only A Game曾经基于游戏对玩家情感的影响作出一个调研，排比出玩家在游戏中的10大普遍情感（1040份调查问卷），依次是有趣（4.28，5分制，下同）、满足感（4.09）、惊叹（4.07）、兴奋（4.02）、好奇感（3.92）、自豪感（3.89）、吃惊（3.59）、欣慰感（3.57）、安慰（3.26）和欣喜若狂（3.26）。对此知名游戏博客Lost Garden曾就《Triple Town》所涉及的情感元素进行了剖析，涉及到期待下一个道具模型的好奇心、创造出美丽景象的自豪感、对碍手碍脚远程传送忍者的厌恶感、对道具在不正当时间出现的愤怒、对自己版图无能为力的无助感、对突然出现神奇移动所带来的期待和欣慰感等，直接将用户的情感灌输在游戏设定的每一个具体的环节中，也正是因为情感的渗透慢慢地将抽象的游戏概念演变为丰富多变的游戏世界，时刻维持着游戏机制与情感节

奏的相互协调，甚至反向互补。

而这也可能正是《Storybricks》作者Stéphane Bura所宣扬的情感工程学的定义，丰富的内涵引导着玩家持续而富有激情地往前探索游戏。显而易见的游戏需求伴随着复杂而多变的游戏解决方案层，既挑战玩家懒散的游戏惯性，又能够在恰当时让玩家在彷徨无助的情况下重新燃起未来的希望，甚至让玩家在这一进程中成为其他后进玩家的先导者，以获取更高的荣耀。

环节2：关于情感与沉浸的论述

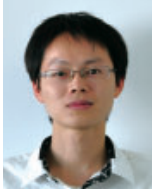
Stratus Technologies设计总监Keith Stuart将玩家在游戏中的真实沉浸描述得淋漓尽致：玩家在游戏中进入了某种沉浸，很快就进入了虚拟的状态，忽略了时间的流逝、没有觉察身边人员的走动、完全与游戏的当时氛围同步。Microsoft Studios用户体验专家Sean Baron更进一步拓展了这种随机意识：玩家本来可能只是打算稍微体验下游戏单纯消磨下时间，但因为游戏设置的关系突然间完全融洽在其间，不知不觉几个小时过去之后才意识到自己正在扮演一个某个游戏角色。

著名研究者Nicholas Yee早先就从更为理论的角度来阐述游戏机制和心理层面的关联影响，这些因素可能包括成就感、人际关系、沉浸感、逃避现实和控制欲。心理学家Mihaly Csikszentmihalyi将之归因于经由游戏设计师合理引导而不断获得强化的潜意识认知，包括我们平时能够在现实生活中感知的，以及脱离于生活但能够随处寻找到合理影像的游戏素材，经由这种题材的熟悉度和玩家在游戏中所架构起的社交临场感达到了直接性和沉浸性的双重感受。

Storm8的Linux系统设计师Geoff Howland将它引申为五种玩家的心态体验，包括完成任务的沉浸感（特别是作出难以抉择的或者随机性很强的决策并最终征服了挑战所呈现出来的感触）、竞争的沉浸感（竞争与协作所维持的长期交互）、控制权的沉浸感（提升玩家在游戏中的自我主导）、探索的沉浸感（让玩家在必要引导的情况下独立去发现游戏）以及获得高分的沉浸感（来自直接的成就）。在最后一个成就沉浸的问题上，游戏设计师Lucas Blair似乎更具话语权，他在定向Angry Birds系列的研究中梳理了完整的成就意识对玩家心态的影响，他认为有价值的成就系统最终可能会影响玩家的游戏进程以及积极性和态度，诸如在《Angry Birds Rio》中的星级评定和New HighScore对于任何一个玩家都具有心理暗示和行动激励（即使是在《Angry Birds Space》中玩家也会执着在如何以三颗星的方式顺利通过游戏关卡，即使当前受限不能也会想尽各种方式再度回到游戏以完成自己对三颗星通关的强烈愿望），或者可以说成就系统更大的功能在于引发玩家的内在动机，从心理分析学的角度看，激起玩家对某些特征属性的沉浸有助于驱动玩家的时间和情感投入，而不仅是一个简单的成就排行榜或可有可无的反馈机制。

基于这情况，游戏插画家Michael Heald认为杰出游戏最大的特征是能够立即引起情感共鸣（投入情感就会立即同其建立联系），好像每个玩家都具有感知的真实性，都能够被这种预设的认知会被带进游戏中，并产生某种背景的熟悉度而与游戏本身出现情感的契合度，游戏分析师Douglass C. Perry认为在这样的情况下就必然意味着玩家已开始认同和珍惜自己同其他玩家或者同游戏本身的虚拟纽带关系。

对于很多顶尖的开发者而言，捕获用户在情感线层面的投入已成为未来必需的层面。Ngmoco公司CEO Neil Young曾多次呼吁开发者需要更多关注用户的游戏体验，甚至将游戏的研发进程比拟为类似电影的制作。按照Ngmoco游戏开发主管Clive Downie的说法，游戏玩家是具有情感和智慧的群体而非简单的付费对象，就像音乐或者影视一样，需要为玩家带来内在的促动价值，特别是情感投入的回报，也只有我们前面提到的社区性的交互和玩家情感投入才有可能最终有持久性的黏着力，独立游戏开发者Steve Swink认为游戏的真正价值在于它让多少人为之动容。



郑金条

游戏邦负责人，游戏邦主要关注和解析国内外社交游戏和手机游戏领域，并定期做深度行业阐述。

责任编辑：陈博（chenbo@csdn.net）

Web App对企业移动开发的影响

文 / 刘铁锋

作者认为，对于传统企业的移动开发，多开发平台带来的学习成本、人力成本以及开发团队的管理成本都不容小觑，而Web App能为其带来很好的解决方案。

短短的几年之间，智能设备的发展已极大地改变了用户的使用习惯，甚至直接颠覆了传统的手机生产厂商。例如手机行业曾经的领袖诺基亚，2012年4月17日被穆迪投资者服务公司将评级下调至接近垃圾级的Baa3，据称主要原因是第一季度手机销售大幅下跌以及低端市场份额遭到其他手机制造商的蚕食。根据华强电子产业研究所的预测，在2012年，中国的智能手机的出货量将会由2011年的4800万台激增至2亿台，占全球智能机市场的30%。而根据美国博客Apple 2.0对48名分析师的调查显示，预计苹果在第二财季的iPhone销量将最多会达到4400万部。根据这样的发展趋势，智能手机将会超过传统功能手机的出货量，并且成为用户日常使用的主要工作手机。而这，也会对IT产业的开发模式和开发方法带来实质性的改变。

移动互联网的主要开发者

归根结底，技术是为解决问题而存在的。毫无疑问，移动设备上的相关开发技术同样也是为解决IT产业的问题而存在。我们可以比较粗浅地把需要使用移动开发技术的公司分成如下几类。

- **移动互联网公司**。移动互联网公司是完全新兴的公司，也是伴随移动互联网兴起而诞生出来的新行业。从2008年开始算起，至今不超过4年的时间。而移动互联网确实对移动互联网技术跟进最迅速，也是被移动操作系统的发展驱动的公司。不管是Android在短短的十年时间从1.5版发展到现在的4.0版，甚至在半年之后将要推出的5.0版，还是iOS从2007年发展到现在5.1的版，移动互联网公司都是一路追踪最新的技术发展，成为行业成功范例最多的公司。不管是刚刚被以10亿美元收购的Instagram，还是刚刚被估值到2.5亿美元、靠超酷界面咸鱼翻身的Path，或是被誉为最强交互实现的Clear。移动互联网公司总是利用移动设备的各种特性，为用户带来意想不到的惊喜。

- **传统互联网公司**。互联网公司和移动互联网其实一脉相承，游戏的规则并未发生任何改变，而仅仅是流量入口从PC转移到了移动设备。而对于传统互联网公司来说，并无所谓Web App以及Native App的负担，因为它们所有的业务都构建在Web基础之上。

- **传统企业**。传统企业是对外界变化响应较慢，但同时又是从业人员最多，开发需求最贴近真实

用户的一个群体。他们会更加稳重，但更不愿意承担技术选型的风险。

因此，根据这个粗浅的分类，我们能够感受到移动互联网开发技术的传播速度：会由移动互联网公司兴起，然后影响互联网公司，最后辐射到企业开发中来。而移动设备的爆发性增长，同时也激发了企业移动开发的需求与发展。

传统企业开发的特点

根据我的经验与认识，可以将不使用IT技术作为主要盈利手段的公司，统一视为传统企业。比如说重度依赖于IT技术的金融业、证券业，以及轻度依赖于IT技术的零售业、制造业等，甚至于政府的信息化需求，这些都可以认为传统企业。这些IT技术的发展主要针对企业内部需求，开发出的系统和产品主要针对企业自身的员工。

那么，这些企业内部的业务系统开发的主要特点如何呢?

- **以MIS系统为主**。企业内部的业务系统，最主要的是依赖于IT技术来做信息的管理。比如大家所熟知的ERP系统、CRM系统、OA系统以及各种和企业业务直接相关的业务系统（比如制造业使用的西门子开发的各种车间管理系统，生产管理系统）。这些最重要的就是信息的输入输出和各种数据系统的应用。

- **以系统的集成为主**。由于企业里面生产企业繁多，一般一个大中型（规模在1000人以上）的企业的内部系统，可能会多达十几甚至是数十个不同的业务系统。因此，如何把信息系统集成，以统一的登录和身份验证系统实现，以及最后以统一的Portal来展现（比如微软的SharePoint），就成为企业内部系统开发的重点。

- **以工作流的驱动为主**。在企业内部，流程的运转和推动，需要层层的审批以及批复。在国内特有的体制下，审批、签署甚至于会签等各种特殊的需求都会直接驱动和影响企业内部系统的构建以及实施。不仅仅是需要工作流系统的搭建，甚至还要配合BizTalk这样的消息队列系统才能够完整实现企业系统的搭建和部署。

传统企业往移动互联网转型耗费高成本

- **以业务的需求为主**。企业内部系统的搭建，往往更多地需要以客户的业务需求来做各种自定义化的工作，因为不同的行业完全不一样。因此在企业系统里面，有部分像SharePoint、BizTalk之类的平台级的产品，也有Dynamic CRM之内的客户端关系系统，但往往要根据客户的业务需要做自定义的开发。

因此，这就是传统企业应用开发的主要模式。在过去20年的发展中，B/S模式因为没有系统升级、部署麻烦的负担（给10000人的企业升级一个软件）而成为主流技术。但在移动互联网时代，企业面临的最大挑战是如何将这些信息系统搬到智能设备上去。是选择客户端的开发技术，还是沿用B/S的网页展示方式，成为一个最大的问题。

传统企业往移动互联网转型的难点

正如大家所知道，企业业务系统开发的难度在于系统的集成，主要的开发逻辑在于业务需求的复杂。而面对移动互联网的兴起，尤其是移动互联网操作系统的分裂，对企业业务系统的开发带来了极大的困扰。

- **多开发平台带来的学习成本**。因为企业系统开发的特殊性，企业业务开发人员的技术需求主要

在于处理数据的交换以及处理各种因数据处理带来的业务逻辑的实现。因此，并不需要特别炫酷的技术，一般都是采用相对成熟的开发技术，保证系统开发的速度和稳定。而iOS、Android、Windows Phone所带来的新语言学习成本、开发框架的学习成本以及开发模式的开发成本，都已成为企业业务系统开发人员的最大障碍。

- **多平台带来的人力成本的开销**。因为多移动平台的存在，因此原先只要使用一种技术统一在Web里面实现的局面被打破了。企业需要的是懂iOS、Android、Windows Phone开发的三类人员，甚至需要把同一个业务需求做三遍。这对企业来说是一件非常痛苦的事情，而同时因为开发人员的短缺，同时会导致企业的用户成本直接增加。

- **开发团队的管理成本**。对于技术管理人员来说，人员和技术的增加，会直接带来管理成本的增加。不仅仅是新技术的学习和把握，还是需要管理移动开发团队的期望值以及技术发展路线，这都为企业技术管理人员来说，带来了极大的要求和挑战。

因此，在移动互联网时代，企业面对移动开发的需求处于非常矛盾和被动的局面。选择进入，成本不可避免地增加；不进入，看起来似乎会落后于时代。

Web App对企业的影响

Web App开发能力以及HTML5技术的发展，正是为企业移动开发带来了非常好的思路和解决方案。

- **开发成本的降低**。毫无疑问，如果可以复用PC的开发技术，同时又能无缝地迁移到移动平台上，对企业开发成本的降低是非常直接有效的。因此，这是企业用户对各种跨平台开发技术非常热衷的重要原因，同时这也是推动跨平台技术以及Web App发展的一股重要力量。

- **减少招人的难度**。由于可以复用PC的开发技术，那么懂业务系统开发的人员的招募就相对容易，这样也会直接减少企业内部开发人员的管理难度。

因此，企业内部的业务开发人员以及为企业提供开发服务的技术开发商，都会试用各种跨平台技术，做各种技术实现方案，来尽可能无缝地迁移至移动互联网平台。

结语

对于企业移动开发来说，最需要的是什么？

我认为，最需要的就是Best Practice。还有人记得经典的三层结构以及各种展示技术的PetShop/PetStore的解决方案吗？这不知道给多少用户带来了遍历以及提供的技术开发的指导。

因此，对于移动开发来说，同样需要类似的东西。我认为需要两个方面的解决方案。

- **纯HTML5的解决方案**。这是一个相对激进的思路。企业可以考虑直接提供移动访问的网站，并且针对移动设备提供非常好的交互体验，从而无缝地由PC往移动设备上迁移。

- **基于类似PhoneGap的跨平台解决方案**。根据Vision Mobile的报道，已有62%的用户了解PhoneGap和Sencha之类的技术来实现的移动解决方案。这是目前最有可能帮助到企业用户的解决方案。

如果有这样的最佳实践出现，Web App将会在企业移动开发市场中占据重要的地位，也会更加快速地推动Web App的发展。

刘铁锋

百纳信息技术有限公司CTO，W3C标准化组织成员。带领团队开发的海豚浏览器在CNET评选的2011年全球最受用户欢迎Android应用中排名第二。

责任编辑：陈博（chenbo@csdn.net）

B2G：来自 Web平台的挑战者

文 / 陈粲然

iOS和Android已成为移动平台的两大帝国，而由Mozilla和著名运营商Telefonica共同开发的新的操作系统——B2G，从诞生开始，就肩负着瓦解苹果和Google垄断地位的使命。

B2G是什么？

B2G全称为Boot to Gecko。众所周知，Gecko是互联网中是最流行的排版引擎之一（Mozilla家族浏览器以及Netscape 6以后版本浏览器所使用），而Boot to Gecko就是一种网络作业环境（Web desktop），该平台的用户界面与应用程序栈完全采用标准化网络技术创建，并且在Gecko HTML页面渲染引擎上运行。

B2G主要由3个部分组成，UI部分叫做Gaia，名字出自希腊语“大地”，也被称作“万物之母”，它是完全用HTML和JavaScript创建的；中层是经过改进的Gecko页面渲染引擎；底层部分叫做Gonk，名字来自一种在20世纪60年代盛行美国的毛绒玩具，包括Linux内核、硬件抽象层（HAL）、电话协议栈以及其他低级系统构件，一开始B2G的系统层是基于Android的native f/w，后来才逐步替换成自己的实现，这部分是C/C++的代码。当然，B2G既不是基于Android的平台，也不会运行Android应用，其所有应用都基于网络，但仍可通过HTML5 cache Manifest或相关API在离线时使用。而运行这个操作系统的设备被Telefonica称为“Open Web Devices”。

战略使命

Telefonica是西班牙电信集团，世界排名前五位的运营商。移动网络是移动互联网应用接入以及传输承载和计费的物理平台，巨大的流量是其主要的利润点，而随着由App Store引领的新模式发展，极大地削弱了运营商的作用，“网络入口”这个曾经由运营商把守的要地，正在被苹果、Google等公司蚕食。

因此，对于Telefonica而言，开发“Open Web Devices”的目标是通过把应用的开发环境和渠道迁移到网页上，来降低主流移动平台力量的一次尝试。如果应用主要都是跨平台开发，那么平台的力量将会被削弱，作为一个将互联网开放和自由作为使命的完全非营利组织，Mozilla是Telefonica实现这个目标的理想合作伙伴。基于Telefonica 和Mozilla彼此间的信任，B2G项目的进展很快。2010年8月，它们开始首次尝试，而项目正式起始于2011年3月，并在2012年2月完成了所有关键核心程序的编写（拨号器、电话簿、短信等）。而据称其UI体验已超过了最基础的Android手机界面。

Mozilla和Telefonica试图从低端市场开始瓦解苹果和Google两大帝国的垄断地位。据称，Telefonica的“Open Web Devices”的硬件标准已能够与iPhone 3G相媲美，且只需1/10的价格。Telefonica希望通过强化低端市场上设备的用户体验，挖掘并占领新的市场。

离成功还有多远？

在过去的十几年内，有超过25个死去或已处于行尸走肉状态的移动平台，据Andreas

Constantinou分析，这些移动平台的“死因”有四个。1. **过高的创立成本**。Symbian的研发成本超过了7亿美元，而其他平台在初始发展的前两三年内，平均花费1亿美元。2. **矛盾的收益模式**。在Google之前，所有的平台都需要开发者支付费用，但却无法为开发者提供有效的收益模式。3. **缺乏网络效应**。App Store模式带来了巨大的经济需求，而在此之前，这一直是很缺乏的。4. **高门槛**。HTC从2003年开始涉足Android平台，而在2008年才推出G1，可见涉足一个平台的时间和精力成本都很高，导致终端厂商不愿承担风险去在一个未成熟的平台上冒险。

对于B2G来说，它是开源的，得到了世界排前五位的运营商的支持，且诞生于HTML5技术更加成熟并得到行业广泛认可之时。但Vision Mobile商业分析师Stijn Schuermans认为，从本质上来说，Telefonica的“Open Web Devices”还不具备赢得用户和盈利的能力。

- 开放降低了开发商的营销费用，但没有为用户提升更高的价值。
- 现在的网页端还不具备优秀的用户体验，也并不是吸引游戏开发者的好平台。
- 在市场成熟之前，Telefonica和终端厂商需要花费数十亿美元培育市场（微软为了进入这个市场每年向诺基亚支付10亿美元）。

在Stijn Schuermans看来，B2G要成功，需要在五个关键点上赢得与其他平台的竞争。

- **大量开放的API**。Telefonica已为B2G贡献了许多的API和代码，但B2G与iOS、Android和Windows Phone的竞争还需长期努力。
- **为开发者提供很好的服务**。一个正在发展的平台，能够激励创新并且满足不同的使用人群，吸引大量的开发者，为他们提供所需的API并搭建有效的应用发布系统。
- **拥有每年1000万手机出货量的市场**。作为世界前5的运营商，Telefonica是实现这点的关键因素，它需要将终端厂商的兴趣变为投资。积极的是，B2G可以适配与其他系统相同的界面、依托相同的内核和函数库，所以其产品推向市场的速度要比Windows Phone快很多。

The Dead Platform Graveyard

Platform	Vendor	Audience	Born	Died
ALP	Access Co	OEMs	February 2007	June 2010
A la Mobile	A la Mobile	OEMs	2006	2009
APOXI	Comneon	OEMs	2002	2006
Aria	Sasken	OEMs	2006	2006
Azingo Mobile	Azingo	OEMs	October 2006	May 2010
Danger OS	Danger	OEMs	1999	February 2011
ELIPS	Open-Plug	OEMs	2004	November 2011
IXI-Connect OS	IXI Mobile	OEMs	2003	October 2006
LiMo OS	LiMo Foundation	OEMs	January 2007	2011
MeeGo	Nokia / Intel	Developers	2009	Zombie
MIDAS	OpenWave	OEMs	November 2005	2008
MMI Solutions	e-SIM	OEMs	February 2004	2008
MOAP	NTT DoCoMo	OEMs	June 2006	2009
MotoMAGX (L-J)	Motorola	OEMs	2001	October 2008
Nokia GEOS	Nokia	OEMs	August 1996	2000
OpenMoko	Openmoko Inc.	Developers	November 2006	2009
Palm 5/6	Palm, Inc	Developers	June 2002	2009
Prizm	MIZI Research	OEMs	2003	October 2008
SKY-MAP	SKY MobileMedia	OEMs	December 2003	2008
SavaJe OS	SavaJe	MNOs	1999	April 2007
Soleus	Intrinsyc	OEMs	December 2006	2008
Symbian OS	Nokia	OEMs / developers	1999	Zombie
TTPCom AJAR	Motorola	OEMs	2002	June 2006
UIQ	UIQ Technology	OEMs	September 2000	September 2008
WebOS	HP	Developers	January 2009	Zombie
Windows Mobile	Microsoft	OEMs	2002	Zombie

source: VisionMobile

Note: Zombie platforms are those in a semi-dead or orphaned state

Vision Mobile列出的25个移动平台的诞生和死亡时间表

- **货币化**。即实现有效盈利，这是建立一个健康生态系统必不可少的一环，相信身为运营商的Telefonica会为开发者和终端厂商提供一个不错的分成方案。
- **销售**。目前还不知道谁将负责终端的店面销售以及向最终用户推广Apps。

而其他的HTML5平台的竞争者，例如Fackbook平台、Google Chrome等早已建立了一个有生命力的生态系统。它们已积累了大量的屏幕适配经验，并解决了发行、零售和货币化挑战。另外值得注意的是，webOS是一个成熟的操作系统，有着优秀的UI和丰富的应用，以及众多优秀的开发者，而Enyo跨平台开发框架为其提供了更广泛的市场，其唯一的问题就在于缺乏硬件厂商支持。随着其开源，webOS在将来有望强势回归。

总结

总地来说，Telefonica和Mozilla明确了集中于低端设备、切断应用和平台之间联系这两大战略。但它们还未证明HTML5技术可以挑战本地应用，Mozilla必须向这个新平台的Web开发者证明它有能力打造一个充满活力的生态系统。Telefonica不仅需要赢得终端厂商的亲睐，最重要的是，它们能够为B2G平台的开发者提供足够的回报。

狂奔的移动端用户体验设计（上）

文 / 石爽

本文讲述了在移动互联网时代，如何在多变的终端配置、网络环境以及混乱的网页内容下，设计出具有优秀用户体验的产品，并分享了在实际项目中的设计迭代经验。

狂奔的移动互联网

2011年底，中国移动互联网用户已突破4亿，相比2010年增长49.2%（来自易观的数据）。

2012年世界移动通信大会将“重新定义移动通信”作为主题，各类型的IT企业都加入到移动互联网的狂潮，并且出现了新的趋势——以前人们只关注终端与设备的性能升级和交互，而现在终端与设备不再是唯一的主角，操作系统、智能、用户体验以及设备与设备间的通信成为关键，移动互联网与人们生活的紧密结合成为话题。

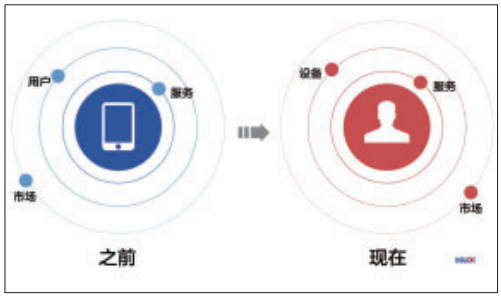


图1　移动互联网与人们的生活紧密结合

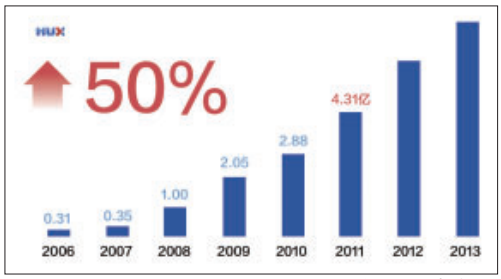


图2　中国移动互联网规模预测

看书、浏览、支付、购物、社交、游戏、下载、音乐、视频、股票、教育等用户生活中的行为都在被移动互联网所改变。面对如此大的机遇，有媒体形容IT企业只有“狂奔”才能夺得广大的用户群和广大的市场——不仅是技术层面的狂奔，用户体验工作更需要狂奔，让信息呈现和人机交互满足狂奔时代的要求。

移动端的用户体验工作，需要站在传统用户体验工作的基础上，结合移动环境的要求——更易认知、更易操控、更易开发、更易扩展，持有积极应变的心态，利用适当的设计和工作方法，高效快速地开展用户体验工作。接下来我们将从多角度详解，如何在移动端开展用户体验工作。

积极面对纷繁复杂的移动环境

纷繁复杂的移动环境，是面向移动端开展用户体验工作需要面临的严峻考验。熟悉这个环境的脾气和规律，才不会被难住。移动环境大体有以下三个特点。

移动状态下的终端，移动状态下的服务

好的设计一定要考虑现实的压力，漏掉某个环节，就会带给用户致命的体验问题。

终端更便携，屏幕尺寸小而多样。我们需要将服务设计得适合用户装进口袋，在手上操控。传统互联网的内容和服务，通过减法设计，做到快速

图3　百度新闻在移动端的呈现都做了相应适配

图4　搜索历史、夜间模式都是为了应对多变的环境

接入、便捷浏览、快速退出。

例如，掌上百度里集合了新闻、贴吧、知道等服务，通过减法设计，使得这些服务在手机上更易浏览。

用户带着终端移动。在移动状态下接入互联网服务，服务会根据用户的移动数据及时做出反馈，与用户互动。例如：用户在终端搜索餐馆，搜索引擎会根据其地理位置的判断而优先给出用户附近的商户信息，并指导用户前往目的地。我们打开百度搜索App时，系统自动判断用户的所在地，给出当地的天气，或者用户订阅的股票走势。信息的基于移动的及时更新，带给用户更加真实的黏性。

移动状态下，环境多变。我们站在十字路口，会因为迷路而查看地图；也有可能在等公交车，随时被拥挤的人群和到来的公车打断；阳光明亮，手机屏幕上的信息看不清楚；手里拎着东西，单手操作不是很方便……移动状态具有较多突发的事件，对我们服务的用户体验影响较多。

夜间模式：手机便携性使得用户在睡前使用服务的可能性增大，熄灯后的手机屏幕非常刺眼，以提供内容为主的应用都会提供夜间模式。我们在设计掌上百度App时，在小黑屋评测多款夜间模式方案，通过多次版本迭代进行完善，可见对用户特定环境的关注。

历史记录：针对移动状态下搜索任务被打断，我们可以想出相当多的策略来应对，例如引入搜索历史记录。这是一个最简单的方法来进行搜索历史任务的管理查询，并快速回到刚才的搜索结果页面。

终端性能参差不齐、网络环境多变

各大巨头的激烈竞争，使得终端日新月异，迭代加速。电池、屏幕、尺寸、系统等存在多样化、差异化。例如网络环境2G、3G并存，4G蓄势待发，Wi-Fi大战正在上演。网络覆盖率因运营商的不同而差异较大，对于用户来说，这个无形的网络还有那么点不靠谱儿。此外，CPU跨度也很大，仅智能机这块，性能差异就非常明显。

面对硬件和网络环境的恶劣环境，我们的设计理念是——高端设计、低端对齐，即在设计方案面向高端机，整合工作兼容低端机，确保中低端机用户体验不做减法，才是完整的体验设计。

例如，掌上百度提供的所有服务都经过云端转码技术压缩，确保用户快速地接收到必要的数据。

在2011年掌上百度以480x320分辨率为标准，开展界面设计，然后再分别优化800x480以及320x240分辨率下的图标和内容排版。而今年我们已将800x480、854x480分辨率作为标准，再分别优化480x320及960x640。确定视觉设计的基准，有利于集中设计资源解决主要需求，避免全面铺开，耗时耗力。

内容格式与质量参差不齐

大部分中小网站都还未做好满足移动互联网用户的工作，依靠转码技术得到的页面或服务显得简陋而且体验差。在移动互联网内容大爆炸的背景下，内容展现需要技术与体验并存的解决方案。

基于环境的这三个特点，设计团队该如何面对呢？我们将设计时经常做的三件事分享给大家。

图5　基于HTML5的页面展现让网页体验大幅提升

图6　成功复用wise成熟的搜索结果内容使研发和设计成本大幅降低

■ **成为移动终端的用户，把玩各类手机终端。**做移动互联网的用户体验设计师，必须要用上Android、iPhone、WP7手机——了解最前沿的移动终端以及操作系统；必须拥有一台长期使用的智能手机——智能手机是未来，长期拥有才能深入了解和体验。有句话说得好，不用奢侈品的设计师，做不出奢侈品。

设计师观察手机激活、设置网络、下载App、使用App、卸载App等一连串的基础服务，欣赏优秀的设计，发现遇到的问题——这一过程本身也是学习、体验、实践移动互联网的过程，这样在开展实际设计工作时，有的放矢，做到照顾全局。

■ **熟读平台设计规范，这是基础教科书。**三年前，对于移动互联网来说，我们都是新人。我们不是平台的设计者，就必须吃透不同平台的环境——这是App赖以生存的环境。目的有二：其一，更快地研发设计用户能用、爱用的服务；其二，分析用户体验的用力点，深入体验设计。以下是一些具体案例。

WebView（网络视图）：WebView能加载显示网页，可以将其视为一个浏览器。它使用了WebKit渲染引擎加载显示网页。我们使用这个系统自带的模块，将搜索结果页面展现出来，成功复用了wise成熟的搜索结果内容，使得研发成本和设计成本大幅降低，也确保了搜索体验在wise和客户端层面的高度统一。

启动图标：图标是用户了解App的关键一环，它是用户与应用互动的第一环节。然而图标会出现在不同的地方，需要不同的尺寸，规范中有详细的列表。我们对App的每一个尺寸的图标进行精修，才能做到品牌的高保真传递。

横竖屏策略：指导原则中有写到，绝大部分手机用户习惯使用竖屏模式，绝大部分Pad用户习惯横屏模式，所以横屏适配在手机上并不是必需的，所以在开发新项目的前期可以暂时不支持横屏适配。

此外，关于声音、通知中心、应用切换、模态视图、基本控件等部分，都能够为我们的产品带来巨大的价值。

■ **鉴赏优秀的应用和设计理念。**学会鉴赏优秀的应用，也就能站在巨人的肩膀上前进。吃透其中的设计理念，融入新的设计项目中。我们非常喜欢品尝新鲜设计案例，阶段性地输出分析报告，深入剖析，并能够在很多项目中融合优秀的设计理念。

Metro UI如实火了一把，带给整个移动互联网一剂新的信息呈现理念。纯二维以及高度概括性弱化了视觉上的复杂性，达到强化信息内容的目的。百度搜索项目组，在搜索结果特性展现的设计时，借鉴了强化内容和信息的设计思路，使用阿拉丁的信息本身进行排版——重要信息更加直接展现，同时满足不同环境的适配需要。未来移动端的百度阿拉丁将更注重效率，整个搜索客户端也偏向效率型设计，降低UI工业化的风格。

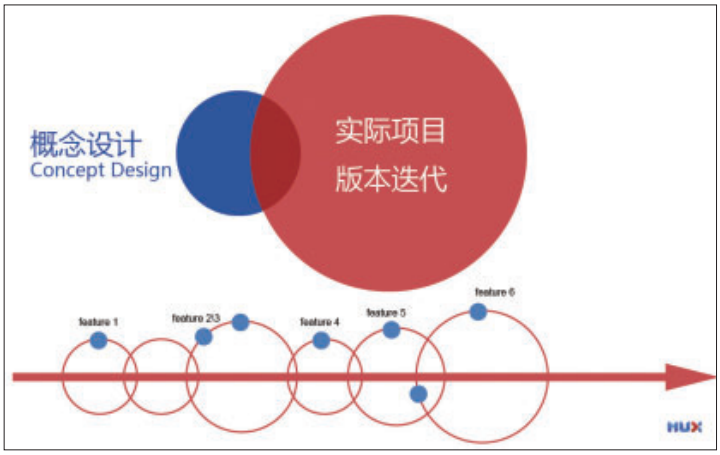


图7　团队需要通过长线和短线工作让产品迭代有的放矢

图8　纸质感及相关细节在百度搜索项目中得到落地和认可

积极面对复杂的移动环境，保持敏锐的嗅觉，洞察移动环境对用户体验的影响，吸收优秀设计思想，这是为产品带来极致用户体验的前提。

设计迭代

移动端产品的设计更加需要快速迭代，难度也很大。传统的PC网站可以快速开展AB测试、小流量测试，从而快速发现问题，并迭代体验方案。在移动端，WAP类产品均可以采用PC站的方法，而客户端的迭代需要开展更多其他的工作。

与用户一起成长

移动互联网产品的迭代比以往迅猛，很多产品都以周为单位发布。而移动互联网的用户更有包容心，允许产品有瑕疵，并愿意期待变得更好。因此，带着用户一起成长，能够在产品和用户两个方面获得双赢。从长线来看，我们需要及时地收集整理用户反馈、积累用户库；短线工作，集中在产品每个版本发布后，在微博、论坛、下载站获得用户的第一感受。通过长线和短线工作的开展，让产品迭代有的放矢。

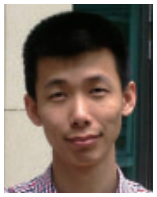
设计前瞻向实际项目输出

设计师一旦深陷迭代快速的项目当中，就会迷失全局的设计感觉。将设计师从实际项目中抽离，结合新技术新观念，对产品进行前瞻的设计讨论和尝试。这种方式，更容易调动设计师的积极性，从而获得横跨交互和视觉的综合产品体验提升方案。设计成果通过团队评估，穿插在实际项目的版本中。

开展前瞻设计工作向实际产品项目输出需要注意以下两点。

- 与当前产品方向和价值保持一致，避免偏离企业核心价值的设计方案；
- 需要了解当前技术储备，不做超现实的概念设计方案。

（感谢百度无线用户体验团队MUX的支持，MUX是业内最早在移动端开展用户体验工作的团队之一，为百度诸多无线产品制订用户体验解决方案，并推动方案在产品中的良性迭代和升级。）



石爽
百度移动・云事业部用户体验部设计经理，从事互联网体验设计工作7年。

责任编辑：陈博（chenbo@csdn.net）

Android系统换肤功能的设计思路

文 / 董红光

作者在文中提出了直接读取外部资源文件、PackageManager、重定向资源ID、重定向资源包路径以及重定向资源文件路径这5种实现Android系统换肤功能的设计思路，并对它们的优缺点进行了比较。

主题其实就是一个满足用户个性化需求的功能。而iOS和Android，作为市场份额最大的两个手机操作系统，其在“主题”方面交的作业是很难让人满意的：iOS官方根本就不支持更换主题，而越狱后的iOS通过非官方方式达到的换肤效果，也差强人意；Android官方声称支持主题，也提供了一系列的方法，一个典型的做法是，在XML中声明一个style，在style中为各种属性赋值，然后在AndroidManifest.xml中为application或activity指定使用的theme，最终可以达到图1的效果。

但从图1来看，Android的主题机制能够提供的可更换元素有限。实际上，Android的这套机制可以支持类如字体、颜色、长宽、间距等界面元素的主题化，但很多关键性的界面元素，例如图片，是不支持主题化的。

而我认为，所有影响用户体验的界面元素，都应该属于主题的范畴，除了上面提到的字体、颜色、长宽、间距外，还应该包括诸如图片、音效、动画等元素。

所以，Android目前的主题机制与我眼中的“主题”还有一定距离。另一个需要注意的关键点是，Android的主题使用，是由应用作者指定的，而非最终用户。这是一个很大的缺憾，用户的选择是多样的，而应用程序只会使用一种特定的风格，所以主题最重要的功能——满足用户个性化的需求，在Android目前的机制上并没有提供。

不过这根本就难不倒勤劳勇敢的程序员们，基本思路很容易想到，拿图片举例，原来程序要用到图A，那么先看外部的资源文件（文中为方便起见，之后统一称外部资源文件为主题资源，对应的包称为主题包）中是否有替代该图的图B，有的话只需修改代码让其使用图B，没有的话就还用图A。那具体到Android该怎样做呢？

Android的资源管理机制

想要改代码，首先需要了解Android的资源管理机制，不过篇幅有限，本文只会对需要涉及的部分基础知识做简单的介绍，深入的探讨可以参考官方文档或其他专题文章。

图1 Android主题机制修改元素有限

首先来看资源类型，Android把资源分成了很多种，比如drawable（图片）、layout（布局）、style（样式）、string（字符串）、color（颜色值）等，而程序使用资源并不是通过路径访问的，而是通过一个特殊的类R.java。应用程序在编译时，Android提供的工具aapt会分析资源，将部分资源编译进APK包的resources.arsc文件中，部分资源经处理后按照原目录结构存放在APK包中，但无论怎样处理，Android都会为每个资源分配一个ID，将该值存在R中，应用程序在获取资源时，提供对应资源的ID即可。例如在colors.xml中定义颜色资源“黑色”：

```
<resources>
<!-- … -->
<color name="black">#ff000000</color>
<!-- … -->
</resources>
```

在编译时，aapt会在R中分配ID：

```
public final class R {
public static final class color {// …
public static final int black=0x0106000c;
// …}
}
```

之后，应用程序只需要通过R.color.black即可访问该资源。

大部分对资源的访问，都被Android封装在android.content.res.Resources类中，而这个类的几个关键方法又多是对android.content.res.AssetManager的方法进行二次封装，后者基本是Native代码，通过JNI的方式来实现。

以上基础知识恐怕就已够我们浮想联翩，各种问号和方案争先恐后浮于脑海了。

不少人的第一个想法是，Android这套东西很复杂，折腾半天连想要的主题功能都不支持，干脆另起炉灶，绕过这套机制，自己管理资源，不就可以“为所欲为”了吗？这种思路当然可行，于是，我们第一个支持主题的思路就新鲜出炉了。

思路1：直接读取外部资源文件

这种方式是指，在代码中，不通过Resources去取资源，而是自力更生，自己管理部分资源的加载，由于很难完全绕过Android的资源访问机制（比如对AndroidManifest.xml和布局layout文件的解析，完全绕过原生重写的成本太高且实现困难），所以在代码中要区分主题化的资源和非主题化的资源。对于前者，自行在外部加载；对于后者，依然走Android原生的机制。并且，如果layout文件的解析依然通过Android原生机制，那么layout中使用到的资源就连带一起被原生机制处理了，这时需要在运行时通过代码来显式更新该资源。

分析一下这种做法：首先，由于需要主题化的资源抛开了Android的资源访问机制，所以主题资源不再需要aapt进行处理了，可以按照自定义的任何方式组织，这无疑为主题制作者降低了门槛，因为不是每个主题制作者都了解如何把资源文件打包成APK文件的。不过对于开发人员来说，就不是一件好事了，因为需要自行编写资源管理相关的代码，这样Android的资源管理机制中所做的各种优化，比如缓存机制、不同DPI资源缺失时的最佳匹配机制等，就统统无法使用了，需要完全重新实现一份，这么做的代价很高。

这种思路另外的一个特点是：程序感知换肤。即程序需要知道某个资源是否支持主题化，以便做不同处理。这样，程序中将充斥着正常界面构建与主题化判断相混杂的代码，很不友好，维护成本也相对较高，不过它的一个优点是，可以控制哪一部分元素支持换肤，不希望被换的部分可以禁止被替换，比如搜狗输入法估计不会希望一个模仿百度输入法的主题把搜狗的Logo也一并替换成百度的，而这种机制就可以避免此类问题。

从上面的分析看，这种思路比较适合应用程序级别的换肤，如果在系统级别做这件事，势必要将所有控件重写。另外，它也做不到为第三方程序换肤，因为换肤逻辑是以代码的方式写在对应程序中的。据我分析，搜狗输入法很可能是采用类似思路实现的换肤。

刚才提到，该思路一个较大的缺点是需要重新实现一份类似于原生资源访问机制的代码，而其成本相对较大，那么对此思路能否改良一下呢？所以下一步可以看看是否能够依然使用原生机制而实现主题功能。通过查找文档和分析Android源代码，聪明的程序员们又找到了一种方式。

思路2：Package Manager

Android提供了一个类android.content.pm.PackageManager来辅助获取应用程序信息，其中有一个重要的重载方法：getResourcesForApplication（String appPackageName）和getResourcesForApplication（ApplicationInfo app），传入一个应用程序的package名称，就可以返回该应用程序的所有公开资源，这样，通过返回的Resources对象和对应的ID即可访问该APK包中的资源。

有了这个利器，对思路1的改进就很简单了：其他部分不变，只是在访问外部资源时，从思路1中的自定义资源组织格式，变成了APK方式组织外部资源，而它带来的最大好处就是不需要重新写一套管理外部资源的逻辑了，因为Android原生就帮助我们实现管理了。

由于其他部分做法不变，所以这种思路同样是程序感知换肤，因此它也只适用于应用程序级别的换肤，而不适用于系统级别和第三方应用程序的换肤。据我分析，GO桌面很可能是采用类似思路实现的换肤。

到现在还没有提到支持系统级别和第三方应用程序的换肤方案。所谓条条大路通罗马，既然在程序代码中实现不了这些，直接改写Android的资源访问代码，对原生机制直接引入主题的支持不就好了？于是各种新的想法又“此起彼伏”了。

聪明的程序员们可能马上会想到一种方法：概念替换。现实生活中也可以有类似的场景，比如有人问我要小米（手机）一代，我故意给他拎一袋儿小米，当然，只要那个人不傻，我是骗不了他的，但当对方不是人而是程序时，就很好骗了，它就是那么信任系统的API，那么这种思路都有哪些方法来实现呢？

思路3：重定向资源ID

最简单的方式就是从ID着手，程序本来向系统（Resources类）要ID为A的资源，系统对其先简单处理，把值A转成值B，然后向下层要ID为B的资源，拿到后假装什么事情也没有发生，一口咬定拿到的就是A，返回给程序。

这样，只需要维护一个很大的ID映射表，当需要A时，转成B，然后一切正常执行，就替换了资源。而在深入了解Android后，会发现一个有趣的现象：系统的资源ID一般都是0x01开头的，而自己的程序一般都是0x07开头的，比如之前用到的系统定义的黑色的ID。

```
public static final int black=0x0106000c;
```

如此一来，对于系统资源完全可以更简单处理，只需把0x07变成0x01就好了，然后主题资源的ID值按照原来系统资源的ID值相应分配，连映射表都不需要了。但对于第三方程序的换肤，还是需要ID映射表的，因为它们都是0x07开头的。

但问题也随之而来，怎样才能保证ID值相应分配呢？其实Android的aapt这个工具默认情况下是随机分配ID的，它不保证两次编译时同一个资源的ID值一定相同，尤其是在资源有增减或修改时，如果想要保证ID值固定，需要给它提供一个public.xml文件，里面指定好想要固定ID的资源名称和对应的ID值。其实在Android的系统资源中，对外暴露的资源也是通过同样的方式实现的ID值固定。

前一个问题解决了，但新的问题接踵而至，既然ID值可能变化，那么映射表就要经常更新，否则给第三方程序做的主题包很可能就会因为资源ID的变化而导致缺失或错误的映射关系。例如拨号这类程序，如果错误了是不能容忍的，比如把09这10个数字图片映射反了，成为9-0，那么用户认为自己拨打了110时可能实际上拨打的是889，而出现“您所拨打的电话是空号”这样让用户觉得诡异的情况。

这时，第三个问题也随之而来：对于系统资源，如果不采取映射表，而采用简单替换0x01这个头的方法，就会导致主题包中必须包含所有系统资源，否则就会出现找不到资源的情况。这样主题包就会变得很大，也是一个缺点。

这种思路可以做到系统和第三方程序换肤，但对第三方程序换肤的支持很麻烦，对系统资源换肤还需要考虑主题包部分资源缺失的逻辑。

可能大家这时会觉得很失落，好不容易找到一种方法，却有很多缺点。我们可以尝试往更深入的地方考虑，如果你已想到“在Android原生机制中，系统是如何知道去哪里找资源”这个关键问题的话，那么很容易就会有了新的思路。

思路4：重定向资源包路径

前文提到，Resources实际上是对AssetManager的一个二次封装，而在AssetManager中，有一个方法addAssetPath（String path）值得注意。实际上，任何一个应用程序启动时，系统都会把系统的资源包（路径一般是/system/framework/framework-res.APK）和该程序自身的APK包通过此方法加入到资源搜索路径中去。因此，如果修改该代码，当加入某个APK包时，其实加入的是另一个APK包不就从根本上解决了此问题吗？所以，这种思路就是，更改AssetManager的Native代码，实现资源包的整体替换。

这种方式看起来很优雅，不过其实和思路3类似，并没有解决为第三方程序换肤时因为ID值可能变化而导致的问题，而对于主题包部分资源缺失的处理，其实要比思路3还要复杂很多，因为原来的资源包已不在搜索路径中了。

看来“釜底抽薪”的方式也不行，那还有什么方式呢？改ID不行，改整包也不行，那改单个资源文件是否可以呢？

思路5：重定向资源文件路径

当程序请求资源时，ID不被修改，但在派发到底层找资源之前，先从某种自定义格式的主题包中查找是否有对应资源，有的话直接解析，返回给程序，没有的话再派发给底层找原始资源。

有人可能会说，这种方法虽然解决了主题包部分资源缺失的问题，但ID变化时依然很麻烦，并且资源解析的操作还需要自己写。

对于第一个问题，由于是自己解析主题包，所以可以完全不通过ID来找主题资源，而是通过资源名称或资源相对路径来建立对应关系，这两者一般情况下是不变的，这样就可以绕过此问题（所以思路1也是不受ID变化影响的）。

对于第二个问题，由于是更改系统代码（Resources等），所以很多底层封装好的方法和变量可以直接使用（思路1就不行了，大部分这些代码对其完全不可见），可以大大降低解析主题包的难度。

其实这时我们已找到一种靠谱儿的思路，以MIUI的主题风格为例，实际上采用的就是思路5。MIUI通过更改关键入口（大部分是在Resources类中）的代码逻辑，将主题机制嵌入到原生机制之中。为了部分原生解析资源代码的复用，主题包也尽量设计得和Android工程（非编译后的APK结构）资源组织形式类似，只是不需要编译，通过资源的名称和相对路径作为标识进行查找。

表1　5种换肤思路对比

思路	主题包格式	程序感知	系统换肤	其他程序换肤	其他
读取外部文件	任意	是	不支持	不支持	
Package Manager	APK	是	不支持	不支持	
资源ID重定向	APK	否	支持	支持	ID变化资源缺失
资源包重定向	APK	否	支持	支持	ID变化资源缺失
资源文件重定向	任意	否	支持	支持	

图2　利用第五章思路实现的主题效果图

董红光

拥有7年工程开发经验，现任小米MIUI系统开发工程师，主要负责MIUI系统的主题功能模块。

责任编辑：陈博（chenbo@csdn.net）

寻找机遇 创造未来 CSDN 热门职位全新推荐

■详细信息请参见CSDN网站

GAMELOFT

历经十年，Gameloft 已发展成为一家国际领先的出版商和视频游戏开发商。从手机到 iPod 再到便携设备、电视游戏机，Gameloft 的游戏遍及全球。

现诚聘以下职位：

- C++ 软件工程师
- Java 软件工程师
- PHP 软件工程师
- 原创游戏策划人员
- 游戏制作人（Producer）
- 2D 游戏美术设计
- 3D 游戏美术设计

简历投递邮箱：recruitmentBJ@gameloft.com
网址：www.gameloft.com

新华网股份有限公司

新华网是由新华社主办的大型网络文化企业，现因事业快速发展诚聘英才。

现诚聘如下职位：

- 高级 Java 开发工程师
- 高级 Web 前端开发工程师
- iPhone 开发工程师
- Android 开发工程师
- 研发工程师（分布式搜索方向）
- 产品设计师

地址：北京大兴国家新媒体产业基地 星光影视园 10 号
简历投递邮箱：dev@xinhuanet.com **网址：**www.xinhuanet.com

新东方教育科技（集团）有限公司

新东方教育科技集团，18 年办学经验，至 2011 年底培训学员达到 1500 万人，业务涵盖早教、学前、小学、中学、大学、在职、出国考试 & 留学申请、英语能力提高、小语种学习、国内 / 海外夏冬令营、图书、在线教育等，旨在为学员提供一站式终身学习服务。

现诚聘如下职位：

- .NET 软件工程师
- .NET 高级工程师
- PHP 软件工程师
- PHP 高级工程师
- 前端开发工程师
- 网站实施工程师
- 软件测试工程师
- 数据分析工程师

简历投递邮箱：zhaopin@xdf.cn，请以“CSDN+ 姓名 + 职位”为标题

ThoughtWorks

如果你**热爱代码如生命**，喜欢有事没事就记录和分享自己的思考与实践，希望远离办公室政治，追求公平合理，追求成为一名优秀的程序猿、攻城狮。那无容置疑 ThoughtWorks 绝对适合你！
你将和一群 Geek **在编码中一较高下**！
你将和优秀**的咨询师**一起，享受在帮助客户解决棘手问题之后的成就感！
你将在不同国家、不同项目、不同技术中**快速成长**！
工作地点：**北京、西安、成都、上海**

现诚聘以下职位：

- Java/C#/Ruby 开发工程师
- 质量保证工程
- 前端开发工程师

简历投递邮箱：job@thoughtworks.com

北京用友华表软件技术有限公司

用友华表自 2011 年起，定位于用友集团 BI 平台、工具及各领域 BI 分析产品的开发和营销。华表公司是目前国内知名 BI 软件厂商，并将在未来两年内打造国际领先的商业智能平台。

现诚聘如下职位：

- 数据仓库技术架构师
- 多维分析技术架构师
- 数据挖掘技术架构师
- Java 高级开发工程师
- VC 工程师
- Android 开发工程师
- iPhone 开发工程师
- Java 集成工程师
- 应用开发工程师
- ETL 架构工程师

地址：中国北京海淀区北清路 68 号 用友软件园 A 座 3 层 1 区
简历投递邮箱：hbhr@ufida.com.cn **网站：**www.cellsoft.cc
电话：8610-62432270

长沙聚游动漫设计有限责任公司

公司的产品定位为游戏机开发，经营理念以市场为中心，充分发挥员工的创造性和积极性，为员工提供一个不断发展的平台，与公司共同成长。

现诚聘以下职位：

- 客户端主程序员：
  5 年游戏开发，2 年客户端主程序经验，主导开发过完整的游戏项目；熟悉一种主流的游戏引擎；使用 C++。
- 客户端游戏程序员：
  具备一年以上单机游戏或网络客户端开发经验；熟悉 D3D 或 OpenGL 的使用，使用 C++。

地址：湖南省长沙市人民路 29 号颐美园 6 栋 1202
简历投递邮箱：csjydm@sohu.com **电话：**0731-82088789

广东南航天合信息科技有限公司

广东南航天合信息科技有限公司是中国南方航空股份有限公司下属全资子公司，是南航集团旗下专业 IT 公司。
目前公司正处于快速成长期，诚挚欢迎各类开发、测试、需求分析等 IT 人才加入我们的大家庭。

加入南航天合，职业生涯从此大不同！

简历投递邮箱：lijialic@csair.com

广州赫基信息科技有限公司

赫基信息科技是赫基（香港）集团旗下分公司， 主要负责集团线上业务及打造电子商务平台。赫基集团为业界精英创造充满前景的广阔天地，提供个性化的福利、透明的培训晋升机制。

现诚聘如下职位：

- 产品经理
- 研发工程师
- 一线支持工程师
- 软件架构师
- 系统工程师
- 运营工程师
- 测试工程师
- UE 高级设计师

地址：广州市天河区林和西路 161 号中泰国际广场 B 座 36 楼
简历投递邮箱：RecruitEC@trendy-global.com
网址：www.trendy-global.com

北京融金广告

现诚聘如下职位：

- PHP 软件工程师
- JavaScript 程序员

更多职位和详细职位要求可登陆 FT 中文网查询！
公司主页：www.ftchinese.com
招聘页面：ftchinese.com/jobs/?from=ft

地址：东三环北路27号嘉铭中心
简历投递邮箱：careers@ftchinese.com，简历请以“职位+姓名+CSDN”为标题

埃菲柯德信息技术（北京）有限公司

中外合资企业，总部位于芬兰的创新之都赫尔辛基。2010 年成为全部研发基于敏捷开发并通过 ISO9001 认证的软件公司。

现诚聘如下职位：

- 销售经理 (5 年 IT 销售经验)
- 客户经理 (3 年 IT 销售经验)
- Android 软件工程师
- iPhone 软件工程师
- Windows Phone 7 软件工程师
- Qt 开发工程师
- S60 研发工程师

地址：北京朝阳区望京地区
简历投递邮箱：hr.china@eficode.com（**标明所申请职位 +CSDN**）
网址：www.eficode.com

杭州海康威视数字技术股份有限公司

“专业、厚实、诚信、持续创新”的海康威视，以人人轻松享有安全的品质生活为愿景，矢志成为受人尊敬的、全球著名的专业公司和安防行业的领跑者。

现诚聘如下职位：

- Java/C++ 开发工程师
- 嵌入式 /DSP 开发工程师
- SOA/SCM 工程师
- SAP 开发工程师
- NOTES 开发工程师
- 图像信号处理 (ISP) 工程师
- 模式识别工程师
- 解决方案工程师

简历投递邮箱：recruit@hikvision.com

浪潮集团通用软件有限公司

浪潮集团是中国知名的软硬件整体解决方案供应商，是我国云计算、物联网、三网融合等方面的龙头企业。浪潮通软作为浪潮集团核心产业之一，是著名的 ERP 软件及服务供应商，在咨询服务、软件开发及解决方案提供和实施服务等方面具有强大优势。

现诚聘如下职位：

- BPM 高级工程师
- Web 应用开发高级工程师
- 数据访问引擎高级工程师
- ESB 高级工程师
- e-HR 系统分析师
- Silverlight 应用开发工程师
- .NET 软件工程师

地址：山东济南高新区浪潮路 1036 号
简历投递邮箱：gshr@inspur.com **网址：**www.inspur.com

寻找机遇 创造未来 CSDN 热门职位全新推荐

■详细信息请参见CSDN网站

完美世界（北京）网络技术有限公司

完美世界诚聘各路英才，欢迎与我们一起开启一个完美世界。

招聘职位：

- 游戏引擎开发工程师
- Java 开发工程师
- Flash（AS3）开发工程师
- PHP 开发工程师

详情请登录 hr.wanmei.com/

简历投递邮箱：duzhipeng@pwrd.com
QQ 咨询：2393130592

聚胜万合信息技术（上海）有限公司

聚胜万合创建于 2009 年初，并在当年获得美国光速创投基金的首轮融资。公司聚合了业界优秀的互联网技术、广告营销、互动创意专家，是专业从事精准营销及数字营销的专业广告技术和服务机构。

现诚聘以下职位（工作地点：上海）：

- 云计算工程师
- 前端开发工程师 /PHP 软件工程师
- 资深系统运维工程师
- DBA
- 前端开发工程师（CSS+JavaScript）

地址：上海市闸北区恒丰路 568 号 恒汇国际大厦 10 楼
简历投递邮箱：hr@mediav.com 网址：www.mediav.com

广州瀚信通信科技股份有限公司

现诚聘如下职位：

- C/C++ 中级开发工程师
- C/C++ 高级开发工程师
- C/C++ 系统架构师
- C# .NET 初级开发工程师
- C# .NET 中级开发工程师
- C# .NET 高级开发工程师
- C# .NET 系统架构师
- 研发项目经理（C/C++ 或 .NET、Delphi 等均可）

简历投递邮箱：yfhr@hantele.com，请以“职位 + 姓名 +CSDN”作为邮件标题

北京法国电信研发中心

Software engineer in the web service system development

Position requirements:

- Master degree in telecommunication, Compute Science OR equivalent
- Experience in J2EE, SOA, Spring, Hibernate, Struts, XML, REST, HTML5 development
- Experience in cloud computing development, such as SaaS, PaaS, etc.
- Strong programming skills, at least 3+ years experience in Java software development
- Familiar with the software development methodology
- Familiar with networking technology
- Good English communication skill
- Good team working spirit and communication skills, and hard working as well

简历投递邮箱：hr@orange-ftgroup.com.cn

网易有道

作为网易自主研发的全新中文搜索引擎，有道搜索致力于为互联网用户提供更快更好的中文搜索服务。依托网易强大的产品服务平台和丰富的资源优势，有道搜索吸纳了众多优秀的创新人才，并正在高速发展中。

现诚聘如下职位：

- 前端开发工程师
- 测试开发工程师
- Web 开发工程师
- 研发工程师
- 有道笔记 / 有道词典 UI 设计师

北京市海淀区中关村东路 1 号清华科技园 3 号楼（创业大厦）2 层
简历投递邮箱：wangjian@rd.netease.com 网址：www.youdao.co

上海征途信息技术有限公司

上海巨人网络科技有限公司成立于 2004 年 11 月 18 日，是以网络游戏为发展起点，集研发、运营、销售为一体的互动娱乐企业。

现诚聘如下职位：

- 资深客户端软件工程师
- 资深服务器端软件工程师
- 资深 3D 引擎软件工程师
- 资深 Java 游戏开发工程师
- 资深 Flash 游戏开发工程师
- Web 前台开发工程师
- 资深 Web 开发工程师（PHP）
- Java 软件开发工程师

地址：上海市徐汇区宜山路 700 号 3 号楼
简历投递邮箱：hr@ztgame.com 网址：hr.ztgame.com

探索Duqu木马身世之谜

Duqu和Stuxnet同源性分析

文 / 李伟，宋凯

同源，是生物遗传学领域的重要概念，用于描述物种或DNA序列是否具有相同的祖先。计算机病毒分析工程师也经常通过分析病毒同源性的方法，追踪病毒制造者的身份——比如dvlodr（口令蠕虫）和lovegate（爱门蠕虫）的制造者，就是通过同源性分析判断为同一人。

背景介绍

2012年3月20日，知名计算机反病毒厂商Symantec在其官方博客中宣布捕获了最新的Duqu木马变种，样本的编译时间显示为2012年2月23日。时隔数月，曾备受关注的Duqu木马又回来了？或者说，它从未消失过。Duqu木马是什么，为什么会这样令人瞩目？原来，Duqu木马一直被认为是此前更为著名的Stuxnet蠕虫的第二代。那么，Stuxnet蠕虫又是什么？

Stuxnet蠕虫亦称震网蠕虫、超级工作蠕虫，具有对工业控制系统的破坏能力，被看做是第一个以现实世界中的关键工业基础设施为目标的计算机蠕虫。有媒体形容它为“超级武器”和“潘多拉的魔盒”，因为它开启了真正意义上的网络战，也为以后的网络攻击形式和恐怖主义行为树立了“榜样”。Stuxnet蠕虫曾于2010年7月暴发，利用微软公司的4个Windows操作系统漏洞（其中3个是当时全新的零日漏洞）、2个WinCC操作系统漏洞和2个有效的数字证书，通过一套完整的入侵和传播流程，突破工业专用局域网的物理限制，攻击用于数据采集与监控的工业控制系统。伊朗政府已经确认该国的布什尔核电站遭到Stuxnet蠕虫的攻击。

Stuxnet蠕虫具有以下四个特点。

- 同时使用了多个零日漏洞。普通病毒作者一般不会这样做（即使是当年肆虐横行的冲击波蠕虫和震荡波蠕虫，也只是使用了一个微软操作系统的RPC漏洞而已）。
- 模块结构复杂，制作周期较长。Stuxnet蠕虫包含了驱动程序、PLC（Programmable Logic Controller，可编程逻辑控制器）指令以及命令与控制服务器（以下简称C&C服务器）通信模块等，普通病毒作者不需要、也不愿意付出这样的开发代价。
- 传播过程复杂而漫长。普通的病毒作者一般急于获得利益。
- 主要用于攻击伊朗的核设施。据发表在德国新闻杂志《Der Spiegel》的一篇文章介绍，该蠕虫感染了伊朗IR-1型离心机后，将其正常运行速度由1064Hz增加到1410Hz，并在15分钟后回到正常

频率，从而造成离心机的损坏。伊朗核计划当局认为Stuxnet蠕虫破坏了约1000台离心机。伊朗也承认其核计划遭到了阻碍，并遭受了“潜在的重大损失”。普通的病毒作者，显然无法由此获得利益。

通过以上分析不难看出，Stuxnet蠕虫更可能是团队制作。知名计算机反病毒厂商Kaspersky实验室甚至认为，Stuxnet蠕虫的攻击是具有国家和政府的支持和协助的。

与Stuxnet蠕虫的复杂相比，出现于2011年的Duqu木马则显得有些单薄，它的主要功能是数据的采集和回传。这些功能似乎并不比常见的QQ盗号木马具有更大的危害，可是，由于它采用了与Stuxnet蠕虫极为相似的技术手段，引起了多个计算机反病毒厂商的密切关注。假设Duqu木马与Stuxnet蠕虫具有相同出处（即两者具有同源性），那么它出现的意义是什么？是否意味着新的Stuxnet蠕虫即将问世？新的Stuxnet蠕虫的攻击目标又是哪种工业设施？该工业设施会属于哪个国家？另外，通过对Duqu木马的分析，是否可以找出Stuxnet蠕虫的作者或者制作团队？

计算机病毒分析工程师通过判断新病毒与已有病毒是否在编写上存在相互借鉴、衍生、复用等关系，找到新病毒的某些线索，最终确定两个病毒是否具有同源性。主要考查病毒样本的以下五个方面：模块结构相似性、编译器架构相似性、关键功能实现相似性、数据结构相似性、病毒作者编码心理特点。

Duqu木马与Stuxnet蠕虫的同源性分析

模块结构相似性分析

Duqu木马的模块结构如图1所示，由驱动模块（使用数字签名）、dll模块和配置文件组成。其中，驱动模块为jminet7.sys，用于解密负载代码和注入系统进程；dll模块和配置文件均以加密形式存储在磁盘上，仅在需要时解密到内存；dll模块的资源中还包含一个负责进程注入的dll模块，因为不需要保存在磁盘，所以没有文件名；该无名模块提供了多种注入方式，通过这些注入方式

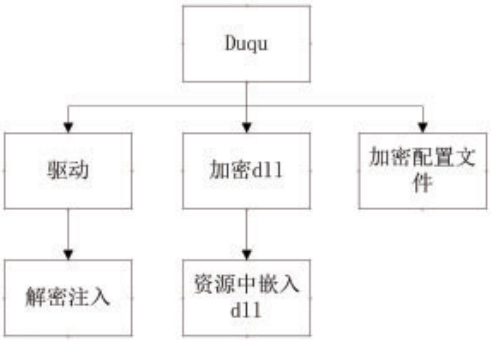


图1 Duqu木马模块结构图

中的一种，将自身的资源节代码（用于与C&C服务器通信）注入到目标进程。

Stuxnet蠕虫的模块结构如图2所示，由驱动模块（使用数字签名）、dll模块和配置文件组成。驱动模块分为Rootkit功能（主要用于躲避反病毒软件）的mrxnet.sys文件和负责解密和注入的mrxcls.sys文件；dll模块和配置文件均以加密形式存储在磁盘上，仅在需要时解密到内存；Stuxnet蠕虫的漏洞攻击代码和工控系统攻击代码均存储在主dll模块的资源节中，在需要时释放出来，主dll资源中包含了用于与C&C服务器通信的功能。

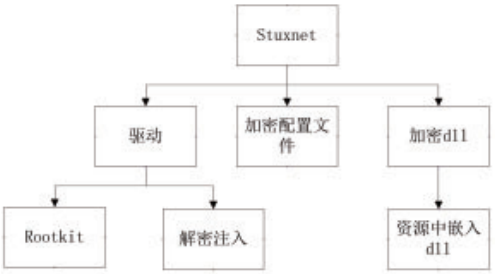


图2 Stuxnet蠕虫模块结构图

通过模块结构的对比，不难看出两者非常相似。两者所使用的驱动模块均采用了有效的数字签名，这一情况在一般的病毒中很少出现。至于两者模块数量的差异是由于两者功能和技术实现方式不同造成的。例如，Duqu木马在Stuxnet蠕虫的基础上又加入了新的注入方法，从而增加了新的模块。

编译器架构相似性分析

Duqu木马的jminet7.sys使用Microsoft Visual C++ 6.0编译。

Stuxnet蠕虫的mrxcls.sys使用Microsoft Visual C++ 7.0编译。

Duqu木马和Stuxnet蠕虫的主dll都使用了UPX压缩壳。

Duqu木马和Stuxnet蠕虫主dll模块资源节中的dll文件都是采用Microsoft Visual C++编译。

很奇怪，后出现的Duqu木马反而使用了更早的VC编译器。

关键功能实现相似性分析

注入方式

Duqu木马和Stuxnet蠕虫都采用了dll注入方式来隐藏自身模块，以躲避杀毒软件的检测，并且都同时采用了驱动层注入和用户层注入这两种方式。

两者都通过PsSetLoadImageNotifyRoutine设置回调函数的方式来完成驱动层的注入，并且两者的回调函数的功能基本一致。每当有PE程序（dll或者exe）被加载，这个函数都会发生回调，从而获得执行机会。如果加载的是kernel32.dll文件，则通过API名称的哈希值，获得kernel32.dll上一些特定的API函数地址。如果被加载是特定的进程名称（该名称以加密形式保存在注册表），则修改此进程的入口点函数跳转到注入代码执行恶意行为，然后跳回到原入口点（很像感染式病毒，不过是完全内存态的感染），成为蠕虫的傀儡进程。

从两者驱动层注入部分代码逻辑结构可以看出，虽然两者采用了不同的编译器（或编译选项），并且在代码中存在少量差别，注册表和主dll文件的解密算法也有不同，但两者的逻辑总体一致，如图3所示。

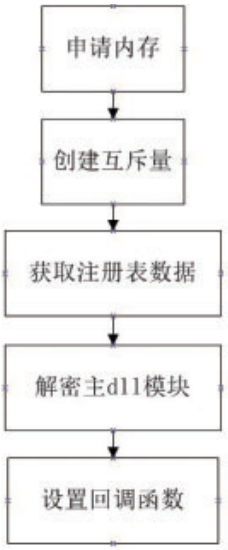


图3　驱动层注入实现方式

通过挂钩ntdll.dll中的下列函数来完成无实体文件的用户层dll注入：

```
ZwQueryAttriutesFile
ZwCloseFile
ZwOpen
ZwMapViewOfSection
ZwCreateSection
ZwQuerySection
```

注入过程为，加载一个特殊构造的dll名称，当LoadLibrary函数加载这个dll时，模拟PE程序加载执行过程，直接从内存中加载这个dll（而不是从磁盘上），从而躲避杀毒软件的文件扫描。图4展示了系统感染Stuxnet蠕虫后，关键进程services.exe中发现的隐藏模块情况。

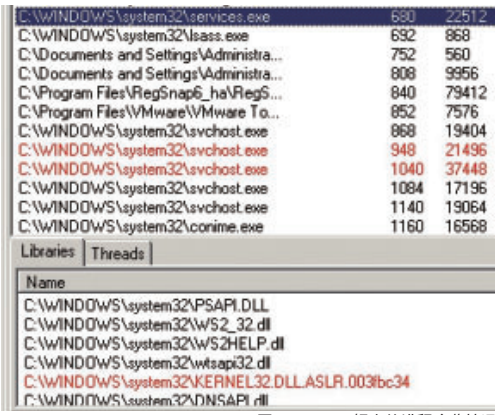


图4　Stuxnet蠕虫的进程隐藏情况

RPC服务

Duqu木马和Stuxnet蠕虫都提供了RPC服务完成点对点通信。图5和图6说明两者使用了相同的rpc绑定方式，分别出现在Duqu木马的netp191.pnf文件及Stuxnet蠕虫的joam7a.pnf文件。

```
10006FB8 sub_10006FB8 proc near
10006FB8 push    ebp
10006FB9 mov     ebp, esp
10006FBB and     esp, 0FFFFFFF8h
10006FBE push    offset aRpcss    ; "rpcss"
10006FC3 call    sub_10006FE0
10006FC8 push    offset aNetsvcs ; "netsvcs"
10006FCD call    sub_10006FE0
10006FD2 push    offset aBrowser ; "browser"
10006FD7 call    sub_10006FE0
10006FDC mov     esp, ebp
10006FDE pop     ebp
10006FDF retn
10006FDF sub_10006FB8 endp
```

图5　Duqu木马的RPC绑定方式

```
100197F1 sub_100197F1 proc near
100197F1 push    ebp
100197F2 mov     ebp, esp
100197F4 and     esp, 0FFFFFFF8h
100197F7 push    offset aRpcss    ; "rpcss"
100197FC call    sub_10019819
10019801 push    offset aNetsvcs ; "netsvcs"
10019806 call    sub_10019819
1001980B push    offset aBrowser ; "browser"
10019810 call    sub_10019819
10019815 mov     esp, ebp
10019817 pop     ebp
10019818 retn
10019818 sub_100197F1 endp
```

图6　Stuxnet蠕虫的RPC绑定方式

关键数据相似性分析

配置文件

Duqu木马和Stuxnet蠕虫都采用了配置与功能分开的实现方式，两者的配置文件结构非常相似。

1. 配置文件的前4字节（被用识别标记）内容相

同，均为0x90,0x05,0x79,0xae。

2. 文件偏移4字节为配置文件头部长度。

3. 文件偏移12字节处为文件长度，区别只是长度的大小不一样。

4. 均包含生存周期。虽然Stuxnet蠕虫配置文件中记录的时间为自我卸载日期，而Duqu木马记录的日期为感染日期，但后者会在感染之后指定的时间完成自我卸载。所以，两者记录日期的目的相同，都是为了控制生存周期，防止分析人员追踪和分析样本。

注册表数据

Duqu木马和Stuxnet蠕虫均由驱动从注册表中读取数据（分别记录在注册表特定主键的Data键和Filter键，均为二进制数据格式），然后实施dll注入。其中，Duqu木马的键名有所变化，但两者注册表的数据结构几乎相同。

Duqu木马在注册表中保存的二进制数据解密后结构如下（包括：dll模块的解密密钥、注入目标进程名、注入目标模块路径）：

```
DWORD control[4]
DWORD encryption_key
DWORD sizeof_processname
BYTE processname[sizeof_processname]
DWORD sizeof_dllpath
BYTE dllpath[sizeof_dllpath]
```

Stuxnet蠕虫在注册表中保存的二进制数据解密后结构如下（包括：dll模块的解密密钥、注入目标进程名、注入目标模块路径）：

```
DWORD control[4]
WORD expNumber;    //注入dll调用的导出函数
WORD Flags;
DWORD encryption_key
DWORDreserved ;
//List
DWORD sizeof_processname
UNICODEprocessname[sizeof_processname]
DWORD sizeof_dllpath
UNICODEdllpath[sizeof_dllpath]
```

病毒作者编码心理特点分析

解密密钥

Duqu木马采用无参数的解密算法解密驱动中的加密数据，得到注册表键值和解密注册表数据的密钥0xae240682，从解密后的注册表数据中获取解密主dll模块的密钥0xae240682。

Stuxnet蠕虫先用密钥0解密驱动中的加密数据，得到注册表键值和解密注册表数据的密钥0xae240682，从解密后的注册表数据中获取解密主dll模块的密钥0x01ae0000。

注册表解密密钥相同，说明该数字对病毒作者有着特殊意义，或者是由于心理习惯使然。

反跟踪手段

在两者的驱动程序中，都有是否需要检测系统状态的判断，而且判断过程是相同的：从加密配置数据中读取参数，与1进行逻辑与运算判断是否需要检测系统安全模式，与2进行逻辑与运算判断是否需要检测系统调试模式。这两处判断的目的是企图避免被病毒分析工程师跟踪、调试。然而，常见病毒样本对抗分析与调试的方法却并非如此——它们会直接判断系统当前运行状态，而不是通过配置参数来决定是否进行这样的判断。参见图7、图8。

程序Bug

从编码心里学的角度看，由于知识体系和编码习惯的原因，由一个作者同一时期编写的代码应该会存在类似的错误（甚至漏洞）。这一特点经常被黑客利用——当有某系统发布了新的漏洞

```
mov     eax, dword ptr byte_15190
test    al, 1
jz      short loc_10611
mov     ecx, ds:InitSafeBootMode
cmp     dword ptr [ecx], 0
jz      short loc_10611
mov     eax, 0C0000001h
jmp     short loc_10632

                        ; CODE XREF: DriverEntry+8D↑j
                        ; DriverEntry+98↑j
mov     edx, dword ptr byte_15190
test    edx, 2
jz      short loc_10630
mov     eax, ds:KdDebuggerEnabled
cmp     byte ptr [eax], 0
jz      short loc_10630
```

图7 Duqu木马判断系统状态

```
mov     eax, dword_13E99
test    al, 1
jz      short loc_1044C
mov     eax, ds:InitSafeBootMode
cmp     [eax], ebx
jz      short loc_1044C

                        ; CODE XREF: DriverEntry+B2↓j
mov     eax, 0C0000001h
jmp     short loc_10460

                        ; CODE XREF: DriverEntry+90↑j
                        ; DriverEntry+99↑j
mov     eax, dword_13E99
test    al, 2
jz      short loc_1045E
mov     eax, ds:KdDebuggerEnabled
cmp     [eax], bl
jnz     short loc_10445
```

图8 Stuxnet蠕虫判断系统状态

补丁之后，黑客会尝试对比补丁定位漏洞位置和类型，然后在漏洞位置附近查找类似的漏洞。经分析发现，Duqu木马和Stuxnet蠕虫在处理注册表数据代码中存在同样的缓冲区溢出漏洞。该漏洞出现在注册表数据复制时，两者都通过ExAllocatePool申请了大小相同（均为0x10字节）的缓冲区，然后使用memcpy函数将注册表数据复制到缓冲区，但在此之前并没有检测注册表数据长度是否正确。如果注册表数据经过特殊构造，则会导致两者出现缓冲区溢出错误。经对比发现，两者代码实现的逻辑几乎完全一样。

Duqu木马和Stuxnet蠕虫在驱动都存在判断操作系统的版本，且逻辑非常相似的。区别在于后者将版本获取函数化，而前者则是使用inline形式执行，这个区别很可能是编译器造成。经过分析发现，两者在判断Windows XP版本时，均将参与比较的MinorVersion和MajorVersion两个域搞反了。这是一个非常低级的错误，很少出现在有经验的程序员的代码中。

总结

主要的比较项目经过整理，如表1所示。

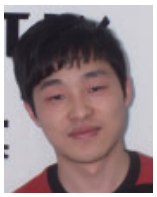
表1　Duqu木马和Stuxnet蠕虫比较

比较项目	Duqu木马	Stuxnet蠕虫
功能模块化	是	
Ring0注入方式	PsSetLoadImageNotifyRoutine	
Ring3注入方式	Hook ntdll.dll	
注入系统进程	是	
资源嵌入DLL模块	一个	多个
利用微软漏洞	是	
使用数字签名	是	
包括RPC通讯模块	是	
配置文件解密密钥	0xae240682	0x01ae0000
注册表解密密钥	0xae240682	
Magic number	0x90,0x05,0x79,0xae	
运行模式判断代码存在Bug	是	
注册表操作代码存在Bug	是	
攻击工业控制系统	否	是
驱动程序编译环境	Microsoft Visual C++ 6.0	Microsoft Visual C++ 7.0

从表中可以看出在模块功能、数据结构、技术实现和编码心理比较中，Duqu木马和Stuxnet蠕虫之间存在很多相似性，尤其体现在相同功能的技术实现方式。不难得出两者均由同一团队制作，或者存在着代码复用（即同源）的结论。

至此，本文通过对Duqu木马和Stuxnet蠕虫的分析和比较，简单介绍了反病毒工程师在分析病毒同源性时使用的方法。其中，通过编码心理学分析病毒同源性的方法由我们在2005年正式提出。值得一提的是，提出该方法的反病毒工程师恰是通过这种方法，得出Sasser蠕虫（震荡波）与Netsky蠕虫（网络天空）作者是同一人的假设，而这个假设在2004年9月德国籍18岁的青年Sven Jaschan被抓获时的供述中得到证实。

其实早在2011年底，Kaspersky实验室已在报告中提出Duqu木马和Stuxnet蠕虫的作者是同一个人（或者团队）的观点，并在另一份报告中提出Duqu木马可能早在2007—2008年之间就已出现，而其主要目的正是收集一系列有关伊朗企业和政府情报机构活动的数据。似乎先有Duqu木马后有Stuxnet蠕虫的假设，更符合实际情况（即先用木马收集详细数据再用蠕虫实施精确攻击），而这也很好地解释了编译器架构相似性分析中的疑点。所以，被认为是Stuxnet蠕虫的第二代的Duqu木马，或许会感觉有点委屈。



李伟

安天实验室安全研究与应急处理中心副总工程师，毕业于吉林大学网络与信息安全专业，目前主要从事恶意代码分析、检测技术以及安全漏洞相关研究。

宋凯

安天实验室安全研究与应急处理中心病毒分析工程师，毕业于哈尔滨工业大学计算机专业，目前主要从事恶意代码分析、检测技术以及安全漏洞相关研究。

责任编辑：卢鹄翔（ludx@csdn.net）

Ready? Go!

Go语言开发背景、语法和类型

文 / 邓楠

Go语言是Google于2009年推出的静态编译型语言，旨在为开发人员提供类似Python、Ruby一样简洁的语言环境，同时又具备C一样的运行效率。Go以社区协作的形式，不断完善语言和标准库的设计与实现，终于在2012年3月28日发布了第一个稳定的发行版本：Go 1。

Go 1的推出，意味着Go语言和它的标准库已经进入了一个稳定阶段。对于谨慎的开发人员来说，开发Go程序正趁当下，现在已经可以放心地使用，不必再考虑未来语法和标准库的变化。

正如Go官方所说，Go 1的目的是发布一个稳定的语言实现，而非全盘修改。所以对于已经熟悉Go的开发人员来说，Go 1与之前的版本并没有很大差异。绝大部分修改都是对标准库命名空间的再组织。本系列文章包含上下两篇，上篇重点讨论Go的开发背景、部分语法和类型系统；下篇讨论Go的并发模型和工具链。

本文的全部代码可以在http://github.com/monnand/goexamples上找到。

编译效率，运行效率，开发效率

曾经那些代码，品读起来，恰似满腹经伦的学者之间，细语轻声，拂琴畅谈。绝非乌烟瘴气之下，面红耳赤地与编译器争辩。

——Dick P. Gabriel

按照Go官方FAQ的说法，Go的出现是为了弥补其他语言在系统级开发上的缺陷。这样一句话，难免会让有些人觉得，Go的诞生纯粹是几位计算机界大佬Robert Griesemer、Robe Pike、Ken Thompson在埋怨自己的不肖后辈，并在吐槽同时，亲自操刀开发的一门语言。但更中肯地说法，恐怕是他们在目睹和体验了Google的系统级开发之后，总结出的一套在异构、分布式多核系统之上的生存之道。而今天Google所面临的问题，也许恰恰是几年后每个公司的面试题目。与其说Go是一座空中楼阁，不如说是各位系统开发界大佬进入新时代后的一部略带辛酸的开发史。

Go的基本设计理念是：编译效率、运行效率和开发效率三者兼顾。使用Go开发，要让开发人员感觉到Python的便利、C/C++的运行效率，以及小到可以被忽略的编译时间。为了实现这个理念，形成了Go语言的以下几个特性。

- 编译，静态类型语言。由此可以提供满足对运行效率敏感的系统级应用。
- 垃圾回收，去除复杂的内存释放工作。
- 简洁的符号和语法，极力减少开发人员输入的字符数。
- 平坦的类型系统，去除了复杂的继承关系。使用结构化类型系统（Structural type system），既简化了事前设计工作，也为未来增加抽象层提供了非侵入式的解决方法。
- 基于CSP模型的并行，简化了并发结构之间的通信和数据共享。为多核时代的程序开发打好

基础。

■ 比线程更轻量的goroutine，让一个线程可以执行多个并发结构。不必使用异步通信，就足以达到线程池与select/poll/epoll的效果。极大简化了多连接的开发。

■ 使用一套简单的规范，开发人员不必再单独编写脚本指定依赖关系和编译流程。仅使用代码本身和Go工具链，就可以处理各种依赖关系。写完代码，一条命令，自动下来各种依赖，直接编译安装。无需make、autoconf、automake、setup.py等工具支持。

前两点应该是不言自明的。本系列文章重点对后5点做详细分析。

化繁为简，语法当先

public static <I, O> ListenableFuture<O> chain (ListenableFuture<I>input, Function<? super I, ? extends ListenableFuture<? extends O>>function)

苍天啊！大地啊！快把这货拦下来吧！

——来自某聊天记录

初学Go，会让人感到它神似C语言，这并非仅是由于其开发团队与C有千丝万缕的联系，也不仅是它们同为系统级开发语言。而是简洁与实用并存的语法让人触目难忘，这里列举部分语法。

■ 没有分号了。其实Go语言规范中，Go中的语句的确是以分号做分割的。但Go语言规定，词法分析器使用一套规则自动地添加分号。这种对词法分析器的要求，带来了一条编码规范：左大括号不要单独写在一行，否则词法分析器可能会产生不必要的分号。虽然多了这样一条编码规范，但带来的结果是开发人员确实不必考虑分号了。以Go标准库为例，其中没有任何一个语句，需要开发人员明确地写下分号。

■ 使用:=操作符声明变量和其初始值，不必明确指明变量类型，因为初始值已经说明了变量的类型。试想要声明一个结构/类。使用Java需要foo.Foo a = new foo.Foo;，而Go则只需a := new(foo.Foo)。对于比较长的类型名来说，这可以减少很多打字。需要注意的一点是，:=操作符同时完成了变量声明和赋值的操作。如果变量之前已经声明，只是要给它赋值，则依然使用常见的=赋值。

■ for、if的判断条件，和switch的控制语句无需括号。也就是说，可以写成if a < b { dosomething() }。还可以写switch b { case 1: dosomething() }。

■ 所有循环只有for一个关键字。而for循环有可以有几种写法: 和C语言for循环类似的写法for i := 0; i < 10; i++ { dosomething() }；和C语言while循环类似的写法for i < 100 { dosomething() }；以及无条件循环for { iteration() }。另外，对于切片类型（类似于Java中的数组，是一种线性结构）和map类型（go中的哈希表实现），还可以配合range关键字，遍历存储的成员，如for i, e := range list { dosomething(i, e) }。

■ switch的条件可以是表达式且可以没有控制语句。这意味着以下语句是合法的。

```
switch {
case i % 2 == 0:
    process_even(i)
case i % 2 != 0:
    process_odd(i)
}
```

■ 直观简单的访问控制。只有名字以大写字母开头的变量、函数/方法、或结构，才能被包（package）外代码访问。否则只是包内可见。这不仅简化了访问控制，而且对开发者来说，只需要看到名字，不必去找声明，就可以知道访问条件。

这些看似不经意的改变，却有效地简化了程序的书写和阅读。让开发者减少打字之外，也让读程序变得更加简单。由于这些改变和特性并不复杂，在此不多做介绍。

正交原则：数据、方法、复用和抽象

早知（C++）如此（复杂），要是能穿越回去，我们肯定会搞一个面向对象版本的C语言出来。

——Ken Thompson，UNIX创始人，Go语言作者

这一节主要讨论Go的类型系统。和C语言一样，

Go也使用结构体进行数据抽象：

```
type Duck struct {
    Name string
}
```

接下来，可以为这个结构定义一个方法：

```
func (d *Duck) Eat() {
    fmt.Println(d.Name,
        "Duck is having dinner!")
}
```

与C++、Java的类不同，为结构体添加方法不必在声明结构体时就声明该方法。只需要保证方法与结构体定义在同一个包（package）中。由此数据与行为被分离，在设计数据抽象（结构体）阶段，不必考虑具体哪些行为。func关键字后面的(d *Duck)表示这个方法从属于哪个类型。d这个变量被成为“接收者”（receiver），在方法的定义中，d的使用类似C++/Java中的this指针。

然后，我们就可以使用这个结构体和它的方法了：

```
func main() {
    d := new(Duck)
    d.Name = "Donald"
    d.Eat()
}
```

按照讲述面向对象语言的规律，下一步应该介绍继承机制了。Go的回答很简单：没有继承。这听起来似乎不可理喻，但却带来了直接的好处：极大地简化了类型系统。一方面，编译器可以更高效了；另一方面，开发人员不必事前考虑各种复杂的继承关系。

但继承的好处也是非常明显的。第一，复用。子类可以直接继承父类的方法的实现，减少了重复代码。第二，多态。开发人员可以使用更抽象的表示（父类）调用具体的实现（子类），由此可以编写通用的代码在抽象层次上进行操作。

Go如果只是粗鲁地去掉了继承机制，而不去面对继承所要解决的问题，显然是不明智的。为此，Go分别使用两套机制来实现继承要达到的效果：匿名成员来实现代码复用；接口类型实现类型抽象。

实现代码复用，Go在结构体定义中，引入了“匿名成员”（Anonymous Field）的概念。了解面向对象设计的开发者一定听说过，要优先使用对象组合而非继承的方式来实现代码复用的原则。Go的匿名成员，实际也就是一种对象组合。

继续上面的例子，假如需要定义一个DonaldDuck类型，它的Eat方法实现和Duck一样，但多了一个叫做Age的成员：

```
type DonaldDuck struct {
    Duck
    Age int
}
```

可以看到，仅仅是把Duck这个类型名字放在结构体的定义中。由于并没有式地给出这个成员的名字，由此得名“匿名成员”。这个写法本质上是定义了另外的一个结构体，它包含了一个类型为Duck，名字也叫做Duck成员（是的，这个成员名和类型名是一样的）。同时，也包含了一个类型为整数，名为Age的成员。

那么这匿名成员究竟对代码复用有什么意义呢？Go对匿名成员有一条特殊规则：包含匿名成员的结构体也具有了匿名成员类型的方法。简单地说，对于上面的例子，DonaldDuck中包含了匿名成员Duck，那么就好像DonaldDuck实现了Duck的各种方法。这样，下面的代码就容易理解了：

```
func main() {
    d := new(DonaldDuck)
    d.Name = "Donald"
    d.Age = 10
    d.Eat()
}
```

我们首先需要申请一个DonaldDuck类型的对象，然后对这个对象的各个成员进行赋值，最后调用Eat()方法。至今为止，我们看到，使用匿名成员可以实现像继承一样的代码复用。但比起继承，它又少了点东西。例如，你无法将一个DonaldDuck类型的对象地址赋值给一个Duck类型的指针。你也不能在DonaldDuck中重新定义Eat方法的实现。简单来说，匿名成员仅仅在语法层面上做了一些简化，并没有触及任何类型系统的内容。对于类型系统来说，DonaldDuck和Duck完全是两个不同类型。

为了实现继承机制的另外一部分，即多态，Go引入了接口类型的概念。与Java中的接口类似，Go的接口也是声明了一组方法的原型，然后由具体结构体（类）的方法来实现各种接口。但与Java不同的是，Go使用了类似OCaml的结构化类型系统（Structural Type System），这种类型的系统不要求实现接口的类型显示地声明究竟要实现哪

些接口。只需要定义好与接口类型一致的全部方法，就说该类型实现了这个接口。具体说来，对于以上代码，我们可以定义一个Animal的接口：

```
type Animal interface {
    Eat()
}
```

这个接口中仅仅定义了一个方法：Eat，它没有任何输入参数，也没有返回值。至此为止，Duck和DonaldDuck都是Animal这个接口的实现。没错，你不必修改Duck和DonaldDuck的代码，不必显式的写明implements Animal，只要实现了Eat方法，并且原型与接口类型中定义的一致就可以了。那么，我们就可以直接声明一个Duck类型的指针，然后把它赋值给Animal接口类型的变量：

```
func main() {
    var a Animal
    d := new(Duck)
    d.Name = "Don"
    a = d
    a.Eat()
}
```

如果还要哪些类型实现Animal这个接口，只需要为这些类型实现Eat方法，就可以了。

这种接口类型为开发人员提供了方便的“先实现，后抽象”的机制。因为接口本身是非侵入式的，不必在定义结构时就明确说明要实现哪些接口。开发人员可以在实现了一些结构之后，再寻找它们之间通用的接口表示形式，进而定义一个接口类型。这时，任何实现了接口中指定方法的类型，都已经是这个接口的实现了。不必再修改以前运行良好的代码，也极大地简化了开发流程。试想，在几万行代码中，寻找几个实现了某些方法的类型，哪怕只在每个类型的定义中添加两个单词，也足以让人生厌了。

这种“先实现，后抽象”的机制也符合我们认识世界的一般规律。哪怕是自己写的代码，往往也是在使用一段时间后，才发现它们背后的一些抽象表示。单纯地要求在代码开发之前的设计阶段就定义好良好的类型层次，是一种无视人类认识局限的荒谬做法。Go作为一种实用的语言，承认这种局限的同时，为人们提供了最小修改代价的方案。

Go的类型系统体现了Go的另一个设计原则：正交原则。线性代数中，正交意味着一组向量之间没有任何一个可以投影到其他向量上。引申到程序设计领域，就是任何特性只针对一个问题，并且互相之间没有交集。具体到Go的类型系统，则体现在方法与数据的分离、复用与抽象的分离。这样的分离使得开发者可以通过分析问题本身，有效地针对问题选择不同特性。

小结

本文重点讨论了Go语言的语法和类型系统，强调了Go的一些基本设计原则。这一切都是以简洁为基础，试图以最小的语言特性覆盖各种常见问题，这酷似当年的C语言。尽管这样的设计也许不会迎来多少学术界的赞誉，却足以把开发者从复杂的语法设计、层层的类型关系中拯救出来，用更清晰简单的方法解决手头的问题。

但只有这些还不足以令Go续写C语言的辉煌。它必须要正视它所处时代正面对的问题，这就是多核带来的挑战。如今的程序，不能再指望仅提高在单核上的运行效率，而提高整个程序的效率。面对多核时代，开发者必须要发掘程序中潜在的并行结构，最大化利用多核的并行能力。而Go则为开发者提供了称手工具，去迎接多核带来的新挑战。下期我们将重点讨论Go语言中对于并行结构的处理与Go的工具链，敬请关注。

邓楠

自由/开源软件支持者，GNU.org中文翻译组组长。一年多以前开始学习Go，目前正在使用Go语言开发一个开源项目http://uniqush.org。

责任编辑：卢鹄翔（ludx@csdn.net）

Windows 8的Metro应用（下）

前两期我们分别介绍了Metro应用的基本特性和支撑Metro应用的新API系统——WinRT。本期我们将介绍如何亲自动手编写一个新的Metro应用，从编译器和编程语言的角度诠释Metro应用。我们使用的开发环境是Visual Studio 11 Ultimate Beta（以下简称VS11），操作系统是Windows 8客户预览版。

主持人：张银奎

《软件调试》一书作者，从事软件开发和研究10余年，对IA-32架构、操作系统内核、虚拟技术，尤其对软件调试有较深入的研究。翻译（合译）作品包括《数据挖掘原理》、《机器学习》、《人工智能：复杂问题求解的结构和策略》、《观止——微软创建NT和未来的夺命狂奔》等。

最小的Metro应用

众所周知，学习编程的一种经典方法就是先用最少的代码写一个可以独立运行的Hello World小程序，然后逐渐加入更多的代码和功能，循序渐进。对于传统的Windows程序，无论是控制台类型，还是窗口类型，都可以用单文件和10行左右的C/C++代码写出这么一个可以独立运行的小程序。但对于Metro应用来说，可没这么简单了。首先，单文件就做不到，因为如我们在2012年3月刊中所说，Metro应用都需要包装，除了源代码文件，至少还需要一个清单文件，以及几个图标文件。其次，因为要使用上一讲介绍的比较复杂的WinRT程序模型，想用几行C/C++代码写出这个程序也做不到。

为了便于理解底层细节，我们选用C++语言；为了控制文件数量，我们暂时不使用XAML。从哪里开始呢？为了避免太多手工劳动，首先还是要利用VS11的项目模板。在VS11中，我们选择Direct2D Application模板，输入MinMetro作为项目名，VS11便为我们创建出一个小型Metro项目。虽然仍有十几个文件，但这已经是预装C++模板中最清凉的一款了。把源文件中的字符串内容（“Hello, Metro!”）和颜色改动一下，然后按F5，VS11会自动编译和执行发布动作，简单操作MinMetro就和创建它的VS11并排站立了，如图1所示。

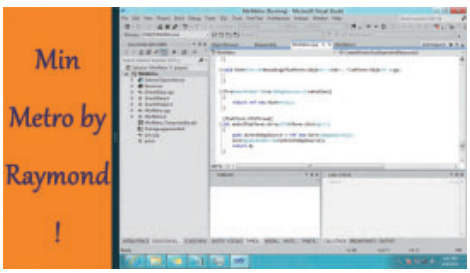


图1 MinMetro和创建它的VS11

文件“普查”

虽然我们尽可能选择轻巧的模板，但实际上创建出的文件仍有不少，表1列出了所有文件的名称、大小和用途。其中MinMetro.cpp是最重要的，它包含了应用的关键类，还有main函数。下面我们从main函数入手来解释MinMetro的内部逻辑。

理解main函数

清单1列出了main函数附近的代码，可以看到main函数中只有3行语句，其中的ref new

清单1 MinMetro的入口代码

```
IFrameworkView^
DirectXAppSource::CreateView()
{
    return ref new MinMetro();
}
[Platform::MTAThread]
int main(Platform::Array<Platform::Strin
g^>^)
{
    auto directXAppSource = ref new
DirectXAppSource();
    CoreApplication::Run(directXAppSour
ce);
    return 0;
}
```

是VC11（VS11中的C/C++编译器）专门为支持WinRT而新创建的，作用是激活WinRT对象，如我们上一期所讲，WinRT是基于COM技术的，combase.dll提供了RoActivateInstance这样的函数来激活（创建）WinRT对象，但考虑到像传统COM那样每次创建对象实例都要调用函数太麻烦了，因此VC11在语言层面帮我们做了很大简化，让程序员在编程时只要像创建普通C++对象那样使用一个new运算符就可以了。但普通WinRT对象和普通C++对象毕竟不同，因此在new前增加了ref以示区别。另外，为了把指向WinRT对象的指针和普通指针区别开来，VC11引入了“^”符号来代表指向WinRT对象指针，读作hat（帽子）。如果使用“……”符号，那么可以把第一句写为：

```
DirectXAppSource^ directXAppSource = ref
new DirectXAppSource();
```

使用C++的auto关键字是为了简化，让编译器自动去推测变量的类型。两种写法是完全等价的。类似的，main函数参数数组中的两个“^”符号也是这样的含义。

VC11中支持WinRT的这些语言扩展被统称为C++/CX，它的地位与用来支持.NET的C++/CLI很类似。熟悉CLI的朋友很容易看出ref new与gcnew何其相似。顺便说下，目前还不支持使用C++/CLI来开发Metro应用。

有些读者可能想知道ref new到底被编译成什么样子。可以用两种方法找到答案，一种方法是跟踪调试，另一种方法是观察编译清单。调试讲得太多了，因此我们介绍第二种方法。首先需要修改项目属性，Properties->Configuration Properties->C/C++->Output Files，然后将Assembler Output项从NoListing改为“Assembly, Machine Code and Source”，点击OK关闭属性对话框后重新编译，编译器便会产生清单文件了，默认是放在中间目录（Debug）中，与源文件同名，扩展名为.cod。打开cod文件搜索main或者源文件行号便可以找到有关的汇编代码了。简单来说，编译器将ref new这行语句编译为如下几个函数调用：

- 调用Platform::Details::Heap::Allocate从WinRT堆上分配一个可以容纳DirectXAppSource对象的内存块；
- 以上一步的返回地址作为参数调用DirectXAppSource的构造函数；
- 调用__abi_winrt_ptr_ctor构建一个WinRT指针，将其赋值给局部变量directXAppSource。

下面继续看main函数的第2条语句，也就是执行CoreApplication类的静态方法Run。

表1 MinMetro项目的所有文件

文件名	字节数	用途
DirectXBase.cpp	12,904	封装DirectX常用操作的DirectXBase类的实现
DirectXBase.h	1,662	DirectXBase类的头文件
DirectXHelper.h	338	只包含一个inline函数ThrowIfFailed
MinMetro.cpp	5,605	包含main函数的应用程序主文件
MinMetro.h	1,665	包含MinMetro类的头文件
MinMetro.sln	2,046	项目集成文件
MinMetro.vcxproj	5,210	项目文件
MinMetro.vcxproj.filters	1,311	描述虚拟目录的项目附属文件
MinMetro_TemporaryKey.pfx	2,460	包含临时签名密钥的密钥文件
Package.appxmanifest	1,294	描述Metro程序包的XML清单文件
pch.cpp	18	支持预编译的C++源文件
pch.h	149	支持预编译的头文件
Assets\Logo.png	5,789	应用的图标文件，150×150
Assets\SmallLogo.png	745	30×30的小图标
Assets\SplashScreen.png	9,381	启动时显示的“片头”图像
Assets\StoreLogo.png	2,005	供网上商店显示的图标文件，50×50

CoreApplication类和App对象

CoreApplication类的全称是Windows::ApplicationModel::Core::CoreApplication，从名字中包含的Core字眼就可以知道这个类的地位不凡，从处理状态变化到管理窗口，再到与UI框架集成，可谓身兼多项重任。CoreApplication类的实例便是MSDN文档中经常提到的App对象（app object）。

WinRT会为每个Metro应用创建一个CoreApplication类实例，也就是App对象。在系统眼中，这个对象便代表着应用程序，是标识和管理Metro应用程序的“把柄”。从应用程序模型的角度来讲，App对象是衔接应用程序的用户代码和WinRT运行时的系统代码的关键桥梁。每个Metro应用只有一个App对象，因此有时又称为“app singleton”。

CoreApplication类公开了5个方法、4个属性、3个事件，如图2所示。

图2 CoreApplication类的公开成员

CoreApplication类的两个Run方法都是静态的，因为App对象是由WinRT的系统代码创建的，所以提供了这两个静态方法供应用程序代码调用。

从实现角度看，CoreApplication类实现在系统模块twinapi.dll中，图3所示的栈回溯便是MinMetro程序启动时main方法调用Run静态方法后，进入twinapi模块，最终执行CoreApplication类的构造函数创建App对象的过程。

App对象创建后，会调用内部的RegisterApplicationServers方法向系统注册并报告启动成功。值得说明的是，如果Metro应用被启动后，没有在规定时间内执行注册动作，那么系统会将其杀掉，稍后我们讨论调试启动过程时还会提到这一点。

两个Run方法的参数承担的角色都是类工厂，供系统代码通过它们来创建用户代码实现的类实例。普通Metro应用大多时候使用的是前一种Run方法（图2中靠上面的部分），其中的IFrameworkViewSource接口用来描述一个可以产生视图提供器（View Provider）的类工厂（源头）。视图提供器的任务就是呈现应用程序要显示给用户的可视内容（视图）。IFrameworkViewSource接口只有一个方法，就是清单1中main方法上面的CreateView()方法，而

```
twinapi!Windows::ApplicationModel::Core::CoreApplication::CoreApplication
twinapi!Microsoft::WRL::Details::MakeAndInitialize<Windows::ApplicationModel
twinapi!Microsoft::WRL::Details::MakeAndInitialize+0xf
twinapi!Windows::ApplicationModel::Core::CoreApplication::Create+0x4d
ntdll!RtlRunOnceExecuteOnce+0x33
KERNELBASE!InitOnceExecuteOnce+0x17
twinapi!Windows::ApplicationModel::Core::CoreApplication::get_Current+0x17
twinapi!Windows::ApplicationModel::Core::CoreApplicationFactory::RunInternal-
twinapi!Windows::ApplicationModel::Core::CoreApplicationFactory::Run+0x13
MinMetro!Windows::ApplicationModel::Core::ICoreApplication::Run+0x32
MinMetro!Windows::ApplicationModel::Core::CoreApplication::Run+0xc2
MinMetro!main+0xd6
MinMetro!_main+0x12f
MinMetro!WinMain+0x8
MinMetro!__tmainCRTStartup+0x22d
MinMetro!WinMainCRTStartup+0xd
KERNEL32!BaseThreadInitThunk+0xe
ntdll!__RtlUserThreadStart+0x4a
ntdll!_RtlUserThreadStart+0x1c
```

图3 构建App对象

CreateView方法的唯一一行代码就是创建并返回实现了IFrameworkView接口的视图提供器（View Provider）对象。

视图提供器

视图提供器是应用程序的界面逻辑与App对象之间的媒介。这个名字的字面意思就是向系统提供可以观察到应用程序内部景象的视图。视图提供器必须实现IFrameworkView。

IFrameworkView接口定义了如下5个方法，从这个接口派生的视图提供器类必须实现所有这些方法。

```
virtual void
Initialize(CoreApplicationView^
applicationView);
virtual void SetWindow(CoreWindow^
window);
virtual void Load(Platform::String^
entryPoint);
virtual void Run();
virtual void Uninitialize();
```

从运行时的时序角度讲，当主线程创建App对象并完成注册后，会着手创建视图，从调试器中可以看到调用的是CoreApplicationViewAgileContainer类的CreateMainView方法。这个方法会创建一个新的线程，我们不妨将其称为视图线程。接下来对视图提供器的创建和初始化都是在新的视图线程中完成的，主要过程依次如下。

- 调用Run方法中传入的类工厂对象的CreateView方法创建视图提供器。
- 调用视图提供器的Initialize方法，参数是一个CoreApplicationView对象，应用程序可以通过访问这个视图对象的事件成员而监听这个视图的事件，比如：

```
applicationView->Activated += ref
new TypedEventHandler<CoreApplicati
onView^, IActivatedEventArgs^>(this,
&MinMetro::OnActivated);
```

- 调用视图提供器的SetWindow方法，通过参数向应用程序“通报”当前窗口。应用程序通常借这个机会注册窗口事件处理器，例如：

```
window->SizeChanged += ref new
TypedEventHandler<…
&MinMetro::OnWindowSizeChanged);
```

- 调用视图提供器的Load方法。目的是让应用程

序在这个时候加载外部资源，做好运行准备，因为接下来就会调用Run方法。

调用第2步中注册的Activated事件处理器。

调用视图提供器的Run方法，通常应用程序可以在这个方法中调用自己的界面绘制代码，开始应用程序的逻辑。例如，在我们的MinMetro中，Run方法中先调用Render方法绘制字符串（到缓冲区），再调用Present显示出来，最后再调用CoreWindow::GetForCurrentThread()->Dispatcher->ProcessEvents(CoreProcessEventsOption::ProcessUntilQuit);目的是防止Run方法立刻返回导致应用程序终止。

通过前面的介绍，可以把Metro应用模型归纳为代表进程逻辑的App对象和代表界面逻辑的视图提供器对象，用一个简单的公式表示就是：

Metro应用 = App对象 + 视图提供器

这样简化之后，main方法就可以简单表示为：

```
auto viewSource = ref new MyViewSource();
CoreApplication::Run(viewSource);
```

这不仅让我们想起了使用纯C#（不用XAML）编写WPF（Windows Presentation Foundation）程序时的Main函数：

```
Application app = new Application();
Window win = new Window();
app.Run(win);
```

二者可谓一脉相承，Metro在app和窗口对象之间引入了视图这个层次，可谓是一个进步，但也是逻辑变得比以前复杂了一层。

推广到XAML

有了以上基础后，我们再来理解基于XAML的Metro应用。还是使用VS11，选择Split Application模板创建一个名为MetroX的新项目。编译执行，然后自动启动调试器（最后介绍），针对图3所示的App对象构造函数设置断点，执行，断点命中后，观察栈回溯，如图4所示。

比较图3和图4，主要的差异是图3中的MinMetro用户代码是直接调用TWAINAPI的CoreApplicationFactory::Run，而图4中的MetroX用户代码是调用Windows.UI.Xaml模块

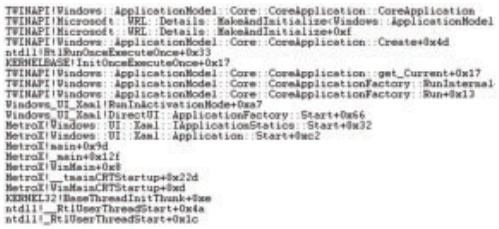


图4　使用XAML的MetroX程序构建App对象的过程

的Application::Start方法将控制权交给XAML库模块，然后XAML库再调用TWINAPI的CoreApplicationFactory::Run。

浏览MetroX项目的项目文件，找不到main方法，而根据图4，MetroX模块中显然有main方法，是哪里来的呢？直接k一下，就有了答案：

MetroX!main+0x9d [d:\raymond\metrox\debug\app.g.hpp @ 63]

看来是编译器帮我们自动产生的。打开这个文件，便可以看到这个main方法的完整面目了：

```
int main(Platform::Array<Platform::Stri
ng^>^ args)
{
Application::Start(ref new ApplicationIni
tializationCallback([](ApplicationInitial
izationCallbackParams^ params) {
auto app = ref new App();
}));
}
```

启用调试

最后向大家介绍一下如何启用调试。首先，如果不调试Metro应用的启动过程，那么可以不做任何准备，只要在Win32桌面运行WinDBG，附加到Metro进程就可以了。但如果要调试我们前面介绍的app对象创建和注册过程，那么使用传统方法（在Image File Execute Options表键下注册调试器）已经行不通了，因为Metro应用启动后，如果没有在规定时间内（数秒）注册，那么就会被杀掉，导致调试失败。如何解决这个问题呢？答案是要借助新的WinRT API来启用调试，这个API就是RoEnableDebuggingForPackage。具体如何做呢？已经有人在下面网站写好了一个，http://winrt.codeplex.com/。关于Win8的系列暂时告一段落，下一期我们将介绍一个神奇的Bug。

本期问题：

上期答案：上一期的问题是RoEnableDebuggingFor-Package函数的作用是什么，答案在上文中已经提到，也就是启用对指定包的调试。启用调试后，会禁止激活超时，并且可以自动运行调试器，更详细的描述可以参阅MSDN中关于IPackageDebugSettings::EnableDebugging的介绍。

本期问题：你知道图3和图4中都出现的WRL是什么意思么？因为篇幅关系我们没有介绍，希望大家自己搜索学习一下。

编者说明：

- 投稿信箱：contest@csdn.net
- 联系方式写在一个单独的TXT文件里，包括以下几项：
  1）姓名
  2）工作单位或学校
  3）电话联系方式
  4）邮寄地址
  5）E-mail地址
- 解答提交时间，最好早于当月15日。

新书上架

本月新书

好学的Objective-C

作者：Jiva DeVoe

译者：林本杰

本书第一部分介绍了Objective-C的基础知识；第二部分深入挖掘Objective-C的功能；第三部分介绍了Foundation框架；第四部分介绍多线程处理、Objective-C设计模式等高级主题。

深入HTML5应用开发

作者：Anthony T. Holdener III
Mario andres Pagella

译者：李松峰 秦绪文

本书分两部分：第一部分介绍的是W3C Geolocation API；第二部分介绍的是使用HTML5、CSS3和JavaScript，利用等轴投影原理开发一款融入社交元素的实时游戏。

Web开发敏捷之道（第4版）

作者：Sam Ruby等

译者：慕尼黑Isar工作组等

本书第1版曾荣获Jolt大奖。第4版增加了关于Rails中新特性和最佳实践的内容。既有直观的示例，又有深入的分析，同时涵盖了Web应用开发中各方面的相关知识。

Java并发编程实战

作者：Brian Goetz Tim Peierls

译者：童云兰 等

本书深入浅出地介绍了Java线程和并发。书中从并发性和线程安全性的基本概念出发，详细介绍了许多设计原则、设计模式以及思维模式，使开发人员更容易构建出正确且高性能的并发程序。

信息安全工程（第2版）

作者：Ross Anderson

译者：齐宁 韩智文 刘国萍

第2版全面更新了第1版的内容，涵盖工程技术基础、攻击类型、专用保护机制、安全经济学和安全心理学等内容。

推荐图书

白帽子讲Web安全

作者：吴翰清

出版社：电子工业出版社

本书分为安全世界观、客户端脚本安全、服务端应用安全和安全运营4部分内容。每一处都很精彩。但由于篇幅有限，本文将仅针对本书第2部分内容进行点评。

第2部分内容讲的客户端安全是近年逐渐走入人们视野的安全问题。

一条简单的JavaScript脚本既可以让服务器瘫痪，也可以窃取大量用户的信息，还可以用作DDos攻击。这不是危言耸听：一个以2的N次方速度传播的XSS蠕虫，可以让服务器瞬间失去响应；一条隐藏在大型网站的持久型XSS，可以劫持用户访问被攻击网站，从而对受害者实施DDos攻击；同样的XSS还可以悄悄地将用户的资料和信息传输给攻击者，让攻击者利用信息进一步发起其他攻击。

上述内容只是传统的攻击方式，本书中还可以看到一些不太常见的示例，如获取客户端IP、Flash XSS；同时除了用常见的截断标签的方式构造XSS，更全面地介绍了其他构造方式，如利用字符编码、location.hash、<base>标签，甚至window.name。

随后介绍了另一种很容易被人忽视的攻击CSRF，一个由小小GET请求构造出的攻击。在看完详细真实的攻击案例后，相信读者肯定会感叹安全的美妙，同时这些案例也一定能引起读者足够的重视。随后针对CSRF的特性，书中详细介绍了有效的防御方式。

点击劫持更是令安全工作者头疼的问题，利用受害者的好奇心或无知，让用户成为攻击传播的一部分。此种攻击更加少见和隐蔽，基本思想就是将一个透明的HTML标签遮罩到正常的标签上，当受害者点击标签时，完成攻击。

在讨论防御手段时也算是相对全面，但我认为有个问题没有很好地解决：有些情况下确实需要将iframe放入第三方网站，但如果这时该iframe遭到覆盖，那么我们是否就束手无策了呢？目前还未发现一个比较周全的解决方案。

HTML5逐渐走入人们的视野，其中的安全问题却鲜有人知。本部分为我们讲述了HTML5中的安全隐患，精练透彻，让读者在感叹HTML5带来更多功能的同时，更看到了随之而来的安全隐患。作者同时对HTML5的安全趋势做出预测，未来的移动端很可能是一片新的战场。

通过研究这些具体的案例，可以让我们更加深入了解攻击，只有懂得了攻击方式，才可以更好地防御。同时在精彩的例子后，作者针对如何防御展开了透彻全面的分析，抓住问题的根源，选择了简单有效的防御方式。

本部分由浅入深，既满足了入门者对攻击的理解需求，又帮助已经入门的安全工作者查漏补缺。不仅是介绍原理，更搭配详细的样例以及作者的观点，让这本兼具普及知识和工具手册功能的书成为当今安全界实属难得的宝典。（**点评人：刘丹，人人网安全中心技术经理**）

一目了然：Web和移动应用设计通识方法
作者：Robert Hoekman, Jr.
译者：段江玲 等
出版社：机械工业出版社

作为软件或Web产品的设计师，我们一直把可用性或易用性作为自己的重要设计目标之一。一款易用的软件，可以同时满足用户的需求和我们的商业目标。这本书便可以告诉我们如何设计一款易用的软件，以及如何评估自己的软件是否成功。作者从软件开发初期开始谈起，一直到软件设计后期，通过诸多案例解释并论证了自己的见解或设计原则。他从“形式追随功能”的设计理论延伸到“why追随what”的战略思路上，指出软件开发需要理由和战略，软件的功能也应“师出有名”、“够用就好”。在设计思想上，Robert也在第1版的基础上做出了重大的突破和改进。他在这本书中除了谈到“以用户为中心的设计”和“以任务为中心的设计”之外，还谈到了“以情景为中心的设计”的理念。这种设计理念也将更直接、更有效地指导我们更好地为目标用户进行设计。

本书的另一个亮点是有关说服性方面的设计原则，Robert举例说明了互惠主义、权威性和社会认同等心理学知识在网站设计中的运用，阐述了说服用户按照设计意愿操作的重要性。如今大大小小的软件或Web公司纷纷创建，鱼龙混杂的软件涌现市场，“酒香不怕巷子深”的观念已经不再管用，要想使用户了解并长期使用你的网站或软件，就要用设计去说服你的用户。

在设计细节方面，Robert举例说明了如何防止错误、如何解决有关错误提示的设计问题，以及设计的一致性、简洁性，如何运用设计元素去表达自己的设计。本书案例丰富，通过这些案例你可以充分理解每一条原则的意义和作用，而且这些原则的实践性很强，可以直接用来评估自己的设计。不管是交互设计师、用户研究工程师，抑或产品经理、数据分析师、视觉设计师等其他软件或Web设计的相关人员，通过学习本书你都可以更加深入地理解优秀设计的原则及作用。

本书第1版上市后，获得的好评如潮。豆瓣上有读者称对本书相见恨晚：“四天时间通读，又重读重点做笔记……给我目前的项目很大启发和帮助，让我在界面设计、软件功能、软件架构及项目开发方式上有了不少新的想法。我将这些想法用于改进原先的项目计划，希望以一种更合理的方式开发出更合理的软件，提升软件的质量。目前的改进是有效果的，改进将持续进行下去。”

本书作者是一名交互设计师和易用性专家。他曾任职于GoDaddy.com、MacroMedia、Adobe等知名公司，现在拥有自己的工作室Miskeeto，一直致力于为广大受众提供优质的用户体验服务。除本书之外，他还著有《Designing the Moment》，该书结合31个真实项目的案例，教会人们如何运用设计来解决问题，揭示了关于Web完美设计的秘密。（**点评人：段江玲，本书译者**）

全球排行榜

Amazon

01 Cracking the Coding Interview: 150 Programming Questions and Solutions
02 HTML and CSS: Design and Build Websites
03 JavaScript & jQuery: The Missing Manual
04 Objective-C Programming: The Big Nerd Ranch Guide
05 iOS Programming: The Big Nerd Ranch Guide, 3rd Edition
06 JavaScript: The Good Parts
07 Don't Make Me Think: A Common Sense Approach to Web Usability, 2nd Edition
08 Turing's Cathedral: The Origins of the Digital Universe
09 Head First Design Patterns
10 The Pragmatic Programmer: From Journeyman to Master

天珑书局（中国台湾）

01 App程序设计入门：iPhone、iPad，2/e
02 OS创意程序设计家：iPhone+iPad跨平台通用，3/e
03 版本控制使用Git
04 ASP.NET 4.0专题实务：使用C#
05 打造安全无虞的Web Applications
06 精通Objective-C程序设计，4/e
07 Android 4.X手机/平板计算机程序设计入门、应用到精通，2/e
08 一定要学会的HTML5+CSS3网页设计实作应用
09 Android初学特训班
10 Google Android SDK开发范例大全，3/e

第二书店

01 白帽子讲Web安全
02 Java编程思想（第4版）
03 Java并发编程实战
04 Linux/Unix设计思想
05 HTML5高级程序设计
06 社交网站的数据挖掘与分析
07 Java核心技术,卷1（原书第8版）
08 编译原理（原书第2版）
09 研究之美（英汉对照）
10 人人都是产品经理

GEEK产品

D-roll笔记本

D-roll其实更像一个装画的画筒。其特点是屏幕和键盘都可以像画轴一样卷起来，跟画筒一样还配有一个皮带，可以很方便地背在肩上，而且它还有一个可拆装的摄像头。

Swiftpoint激光无线鼠标

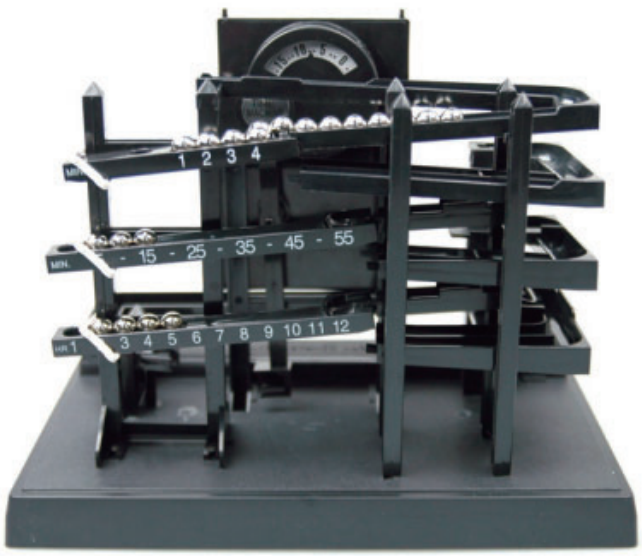
这款史上最小的鼠标来自新西兰的科技公司Swiftpoint，它虽然体积极其微小，但拥有超长的电池续航能力、便捷的滚动和缩放功能和1000Dpi的精准度，90分钟的完全充电可保证2～4周的鼠标正常使用。

TIWE手表

这是由中国设计师吕中方带来的TIWE手表，它的灵感来自满天繁星。整个OLED表盘屏幕上，布满了大大小小的白色圆点，它们都在随机运动着。当你摇动手表时，这些小圆点会自动组合成分钟和时钟，显示出当前时间。

iPhone宠物狗

日本玩具生产商Bandai宣布推出以iPhone作为大脑的宠物狗。它是通过一个免费App来进行控制的。对于主人在其“脸上”触碰会有超过100种脸部表情进行反应，同时主人还可以通过声控或者前置摄像头进行互动操作。宠物狗会不断学习更多动作，还能通过蓝牙装置与其他宠物狗进行歌唱表演，它本身还是一个闹钟和免提装置。



Time Machine Tabletop Clock

这款被称为时光机的桌面计时工具让计时变得十分有趣！只要打开开关，络金小球就能准确地计算每一秒钟，跑一圈刚好是一分钟。设计者的想法是让人们可以清楚地看到“时光”在自己面前一秒秒地流走。

概念手机Transducer Mobile Phone

不管对方说的是哪国语言，都能通过本手机内置的翻译机自动翻译成用户听得懂的语言，还能在光滑的彩色屏幕上显示出来。除此之外，Jasper Hou设计的这款手机还具有GPS功能、透明的彩色LED和呼吸灯。该手机还具备数字扫描功能，通过手机上方摄像头扫描到的图像、文字、书本、海报、菜单等翻译并转化成语音。

AWS推动者

Werner Vogels

文 / 高松

Amazon云计算平台AWS（Amazon Web Services），为很多公司提供了基础云服务。Dropbox、Instagram、Quora、Foursquare、Reddit、Heroku这些互联网新星能耀眼上升，AWS居功至伟；它让有想法、有技术的小团队，以低价获得世界级的运营服务，负担得起用户迅速增长的压力。既然AWS如此重要，那么不得不提及Amazon CTO Werner Vogels——AWS主架构师之一。

Werner Vogels在1958年出生于荷兰阿姆斯特丹。在接触到8位计算机Atari后，他对计算机科学的兴趣便一发不可收拾。Vogels早年在荷兰海牙应用科学大学读书，彼时PC尚未普及，他笑称能使用只有16位的PDP-11小型机已算幸运。

20世纪90年代初，30岁出头的Vogels来到葡萄牙里斯本，在波尔图大学系统与计算机工程研究院（INESC）任高级研究员。“那时在学术方面尚无建树”，在同事的引领下，他加入开发团队，不断完善自己的技能，发表了首项科研成果。

1994年，Vogels远赴美国，在康奈尔大学计算机系做了10年的研究工作，主要领域是具有扩展性的可信赖企业系统。在那里他首次尝试与团队合作，在非常有限的时间内构建了U-net，一个用于并行和分布计算的用户级网络接口。那次合作给Vogels留下了极为美好的印象，绘就了他日后职业走向的蓝图。

1999～2002年，Vogels在Reliable Network Solutions公司，兼任副总裁和CTO。这期间他开始写博客《All Things Distributed》，此时他已成为顶级的分布系统专家。在自己的博客里Vogels撰写了大量的分布式领域文章，以往只存在于学术讨论中的概念得到了极大的传播和普及。他还写过很多参考资料和专栏文章，关注领域也是分布式计算技术。

2003年，Vogels获得阿姆斯特丹自由大学计算机博士学位，两位导师Henri Bal教授和Andy Tanenbaum教授也鼎鼎大名，分别是著名的类Unix操作系统MINIX的作者和并行计算领域专家。

至此，我们看到领域专家知识的积累过程，他去过不同的国家，与不同文化背景的团队合作，做过最底层的技术，也做到了技术管理的高位，既在学术钻研有所造诣，又兼顾商业实现，他对自己瞄准的领域一以贯之，历久弥坚。

2004年Vogels加入Amazon，任研发部门经理，2005年初被任命为CTO兼副总裁，并开始负责Amazon全球的架构设计，以及包括云计算在内的技术创新。Vogels在公司内外有着重要的地位，是除了Jeff Bezos外唯一可代表公司的对外发言人。

加入Amazon后，Vogels的博客写作也开始转向，从之前主要谈论自己的研究成果，到更多地以产品为导向，介绍更普及的技术和工业理念。除此以外，转向也体现在他开始更为看重团队合作。

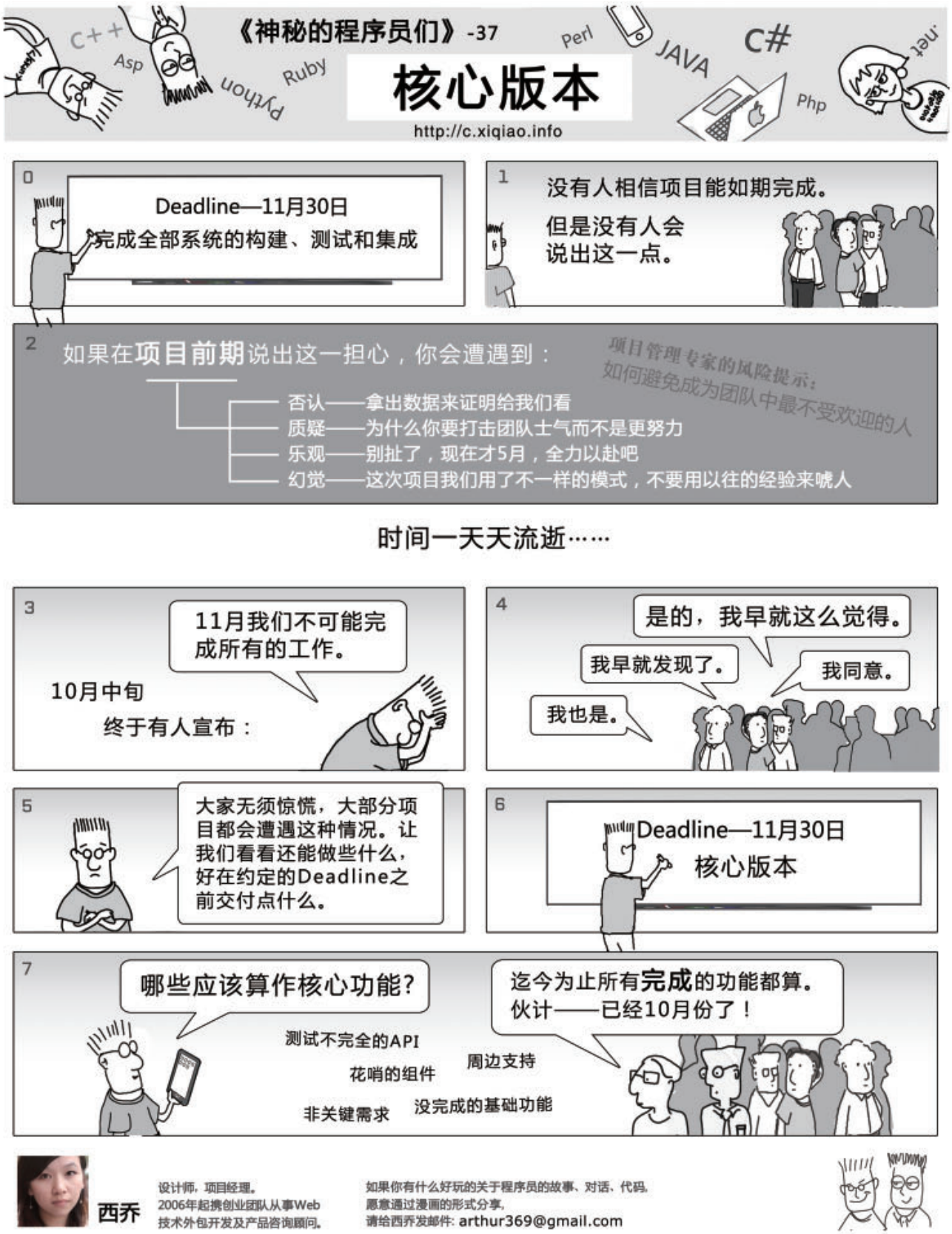
2008年对Vogels而言是标志性的一年，他成为AWS架构师之一。《Information Week》杂志因Vogels在云计算方面发挥的普及和推动的贡献，将年度CTO/CIO奖项颁发给他。

Vogels并不信任通用平台，即一个平台到处适用，因为每个平台都只能满足部分开发者、应用和用户需求。Vogels一直强调构建AWS工作的核心是鼓励创新，他经常与AWS上的应用开发者沟通交流，及时宣讲他们的创新成果，帮助搭建“平台上的平台”。

Vogels曾在一篇论文里深刻表述了Amazon架构技术，介绍了专门为了购物车设计的存储引擎Dynamo，此篇论文获得OSDI最佳论文奖，此后这一引擎被应用在Amazon的S3上。

最新网络流量分析显示，Amazon处理了1%的互联网流量，1/3的网民每天都会访问AmazonAWS托管的网站，Amazon云计算正成长为互联网的核心组成之一。

成文之际，Werner Vogels在博客中宣布了AWS新服务Amazon CloudSearch，正如他所言：“云计算才刚开始。”



西乔
设计师，项目经理。
2006年起携创业团队从事Web技术外包开发及产品咨询顾问。
如果你有什么好玩的关于程序员的故事、对话、代码，愿意通过漫画的形式分享，请给西乔发邮件：arthur369@gmail.com